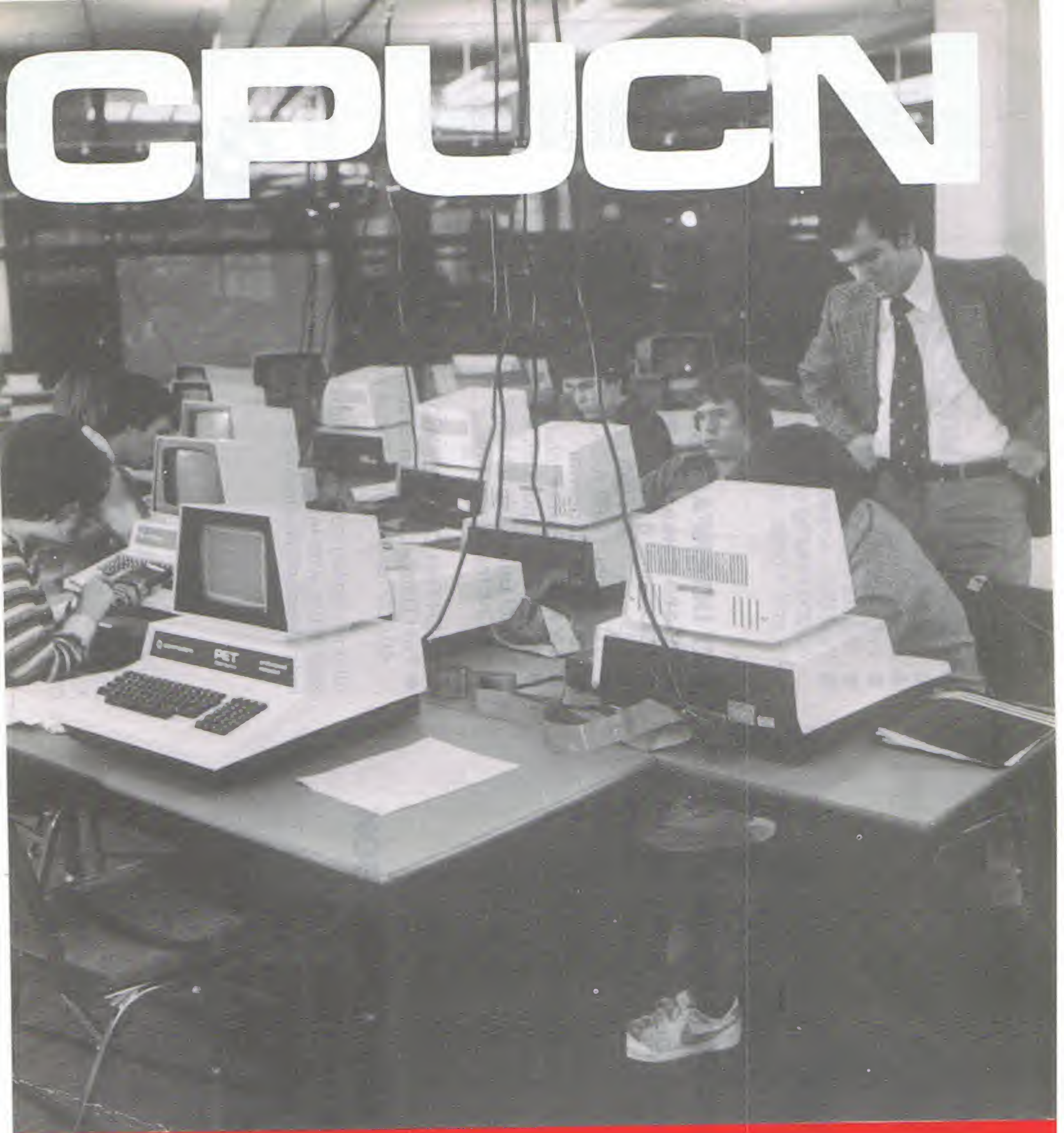
CPUCN



Volume 3
Issue 2

commodore

CONTENTS

EDITORIAL

Dave Middleton

It would appear that more and more people are getting into machine code programming, I receive far more machine code than BASIC. This leads to a problem. Machine source code by its nature tends to be very space consuming especially if it is going to be understood by others. I could of course just print a hex dump which can then be entered into the PET but this is unsatisfactory on two accounts. Firstly it is very difficult to understand both how the program works and even more difficult to change it so that it can be made to perform a different task. Secondly, a well documented assembler listing can help others, who do not have the expertise, to start to write machine code. I would welcome your comments on this; do I continue to give a large portion of the magazine over to machine code listings or would you be satisfied with hex dumps or BASIC loaders?

I am suprised by the lack of interest in the Pascal Users Group, is Pascal so time consuming or interesting for people who have purchased the language that they are now hidden away in dark corners gibbering quietly to themselves?

In the last issue of CPUCN there was an advertisement from 3D, (Digital Design and Development) who produce all sorts of interfaces and D/A equipment. The ad was on page 46 and had very poor quality reproduction, 3D were understandably annoyed and for this I apologise as the ad in no way represents the quality of 3D equipment.

I have published some of the letters which I have received and given my replies along with the letter. At least this way they get answered!

The cover shows the first prototype Mu-pet system being operated at Sheridan College. The Mu-pet is distributed in the UK by Kobra Microsystems.

All of us at Commodore wish dealers and users a Happy Christmas and a very prosperous New Year.

JOIN A FRIEND FOR CHRISTMAS!

Do you lend your copy of CPUCN to friends? Do you ever see it again? Would you like to see it again? Is it in the same condition? If the answers are yes/no/yes/no then why not give somebody a good present for the New Year, it's the same price as a cassette of games software but will give lasting enjoyment through the year and at the same time why not give yourself a copy of 'Best Of CPUCN', it contains all the information held in the first two volumes of the magazine and should be a real help to the beginner and expert alike. Just fill in the form below.

HOW MUCH DOES IT COST?

The subscription fee for the User Club is £10.00 for the UK and £15.00/year for overseas and you get free entry to the Commodore PET Show held once a year in London.

SPECIAL OFFER TO NEW PET OWNERS!

As a special incentive to new PET owners the membership fee has been reduced to only £7.50 for UK membership and £12.50 for overseas membership but you have to send us a photocopy of your purchase invoice and it must be received by us within 30 days of purchase.

HOW DO I APPLY FOR MEMBERSHIP?

Simply send a cheque/P.O for £10.00 for within the UK or £15.00 if you live overseas. Remeber you get a £2.50 discount if you send a copy of your purchase invoice. Any appropriate backdated issues will be sent immediately and thereafter you will receive subsequent issues at regular intervals.

Send your cheque to:-

CPUCN Subscriptions
Commodore Business Machines
818 Leigh Rd.
Slough
Berks

MEMBERSHIP FORM

Name______________________________
Address____________________________
____
____
____
____
____

I enclose a cheque to cover the following items:-

Membership fee (UK)	£10.00	_______
Membership fee (Overseas)	£15.00	_______
Best of CPUCN (UK)	£10.00	_______
Best of CPUCN (Overseas)	£15.00	_______

I enclose a cheque and a copy of my purchase invoice thus claiming my £5.00 discount to cover the following items:-

Membership fee (UK)	£ 7.50	_______
Membership fee (Overseas)	£12.50	_______

Total enclosed £___________

COMMODORE NEWS

Dave Middleton

Commodore in Europe

The Commodore PET is Europes leading micro computer and to demonstrate our commitment to Europe we are establishing a production facility in Braunschweig, Germany. It is going to be used for the manufacture of PETs and peripherals mainly for the European market. The new location in Germany was chosen after coonsideration of many factors including the reputation for building high quality technical products.

We are also considering the possibility of establishing a chip plant in Europe to complement the two chip plants we already have in America.

Commodore expands in Slough

The company is in the process of taking over part of another building next door to 818 Leigh Road. The Information Centre at Euston Road is going to be closed down and many of its activities moved to the new buiding along with the Training Department. The offices at 818 are already overcrowded so some of the departments will be moved to give more room generally.

The VIC has being released in Japan. Why Japan? It may be a suprise but Commodre is a truly international company with facilities in Germany, Hong Kong, Japan, Canada, England and of course the USA. Since a considerable amount of the development was carried out in Japan, it was appropriate that the product should be test marketed there. It goes without saying that the Japanese home electronics market is one of the most competitive in the world.

Communications News

The Commodore 8010 Modem has been releaed and will be available from your local Communications dealer. The modem allows PET to PET, or PET to mainframe communication and can originate or answer in full or half duplex mode. At present the modem is a fairly rare beast but it should be available in volume during the 1st quarter of 1981. The cost is £255.00 + VAT.

Having bought your modem you will now need software to enable your PET to talk to to a mainframe efficiently. The software Communicator 1 allows the PET to communicate with DEC mainframes working with the VT52 interface standards, with asynchronous RS232 link at up to 4800 baud, allowing the PET to send or recieve files. Communicator 1 costs £250.00 + VAT and is available for 3000, 4000 and 8000 series computers.

Comlink allows high speed communication via the IEEE bus using a set of efficient machine code routines directly from BASIC. For instance the command CONVERT will change PET ASCII to true ASCII. TIMEOUT is very useful in that it controls the length of waiting time between character receipt or transmission before the PET gives a 'timeout' status. Comlink will be available in the 1st quarter of next year and costs £200.00 + VAT.

Please note that only Communicator 1 is available 'off the shelf' at present but if you would like a some more information about the latest communications products then please write to us at Commodore Slough asking for the 'Commodore Communications' leaflet, you will also be notified of you local dealer.

Also under developement in the Commodore labs is a computerised cash register, no other details are available yet but it should be available in the second half of 1981.

The Commodore 8000 series PET

With the last edition of CPUCN you should have recieved a copy of 'Commodore News'. This contains a description of the new software available for the 8000 series machines. There are three management packages available for the PET these being THE ACCOUNTANT, which consists of 4 independent packages called: SALES CONTROLLER (Sales Ledger with up to 1200 accounts), PURCHASE CONTROLLER (Purchase Ledger, 1200 accounts), BUDGET CONTROLLER (Nominal Ledger, 600 accounts) and LINK which provides an automatic link between the sales/purchase ledgers and the nominal ledger.

COMMODORE APPROVED

Andrew Goltz

The "Officially Approved by Commodore" logo is appearing more and more frequently in advertisements in CPUCN, as well as commercial publications like PRINTOUT, PCW and PRACTICAL COMPUTING. But what does it really mean when a product is "Comodore Approved"? Let's follow the progress of a typical application for Commodore Approval and find out.

I will first ask the applicant to provide full written specifications of the product he proposes submitting for Commodore Approval. These are then considered in collaberation with Dave Briggs, Commodore's Technical Support Manager, and Dave Middleton who, as well as editing CPUCN, looks after "Approved" systems software and programmers' utilities.

If it seems likely that the new product is a significant enhancement to the capabilities of the Commodore PET, the applicant is then invited for an interview at Commodore, Slough. He is usually asked to bring the prospective Approved Product with him for demonstration and will have a thorough grilling on his company's finances, his proposed distribution and marketing strategy, as well as on the technical specifications of his product.

Quite a few would-be "Approved Products" "bite the dust" at this stage and in some cases the company concerned is recommended to collaberate with an existing Approved Product supplier in order to spread the costs of advertising and other promotional activities over more than one product.

If the applicant and prospective "Approved Product" survive the interview, the next step is for a number of the products to be sold to end users - and then to wait.

After a suitable period of time has elapsed, (this could be three months for new accounting software), the applicant is asked to arrange a visit to a number of end-user reference sites. These field visits play a crucial part in determining whether a product can really stand up to a harsh "real world" environment, is properly debugged and, in the case of accounting software, properly carries out the "month end" and "quarter end" procedures.

After the site visit, the applicant may be asked to carry out some modifications to the product, in which case he will be again asked to provide new reference sites. However, if all goes well, the product is "signed up", contracts are exchanged and the new Approved Product takes its place in the next edition of Commodore's Approved Product catalogue.

The following new products have been Approved by Commodore since the last catalogue was published:

Visicalc
ACT (Computers) Ltd
Radclyffe House, 66/68 Hagley Road,
Edgbaston, Birmingham
Tel: 021-455 8585
Contact: Matthew Wauchope

WordPro
Professional Software Ltd
21 Station Road,
New Barnet, Herts.
Tel: 01-441 2397
Contact: Bob Webb

Book-keeping for Solicitors
Anchor Computer Systems
62 Market Street
Holyhead, Gwynedd
Tel: 0407 4520
Contact: Geoff Ingram

Interfaces for Control and Monitoring
ICI Gammatrol
PO Box 1
Billingham, Cleveland
Tel: 0642 553601 ext. 3752
Contact: Ray Broadbridge

Shaft Encoder complete with PET Interface
Cetronic Ltd
Hoddesdon Road
Standstead Abbots, Ware, Herts.
Tel: 0920 871077
Contact: Stuart Timms

Chromadaptor
Sadektronics Ltd
1 North West House, 45 West Street,
Brighton, Sussex
Tel: 0273 29949
Contact: Dr. S. Sadek

PET Desk
Aquatech Composite Materials Ltd
Pear Tree House, Woughton on the Green
Milton Keynes MK6 3BE
Tel: 0908 679528
Contact: Brian Hogan

Supercow
Upthorpe Computer Programs Ltd
Aston Tirrold, Didcot
Oxon OX11 9EW
Tel: 0235 850747
Contact: Mr. D. Hotz de Baar

Telex Prep, Tape Prep, Printer Stand, Forms Receiving Tray
Taylor Wilson Systems Ltd
Oakfield House, Station Road
Dorridge, Solihull, West Midlands
Tel: 05645 6192
Contact: Sandy Livingstone

REVIEWS

Dave Middleton

A review of WordCraft 80

In the CPUCN 3.1 I gave a review of WordPro4 which has been written for the 80 column PET. In this issue I am going to give a review of WordCraft80. Both of these programs are very powerful and differ in their operation slightly. Both are Commodore Approved Products.

I will say at the outset that this is the first time that I have used WordCraft, CPUCN is normally produced with WordPro, the main reason for this being that WordPro was the once a Commodore product.

WordCraft comes on disk with a manual in a white ringbinder, the manual is very clear and consists of 83 pages. There is a protection device which plugs into the cassette port to prevent the purchaser from being able to duplicate the program for friends.

There are three modes of operation for WordCraft:-

Command mode where system controls such as disk initialisation, printing, getting text from disk and saving finished documents back onto disk are performed.

Control Mode where the user inputs the commands which are going to affect the output of the text. For example a command to start a new line or indent text.

With Type mode the user is allowed to create or modify a document in the PETs memory. The user has full cursor control and can move around the screen at will.

A very powerful feature of WordCraft is its immediate formatting. As the document is typed in it is formatted according to the instructions which the user has given it. As you can see this text has been written to a 40 column width. As I was writing it WordCraft took care of moving the word which exceeded the right margin onto the next line. This text is right justified but WordCraft will only perform this task when text is being output to the printer. A feature of WordCraft which is very unusual even on systems costing thousands of pounds is its ability to pan over a document which exceeds the width of the screen. This has obvious uses in laying out large tables. For example, say I wanted to lay out a table which has an output size of 100 columns, to perform this task on the 80 column screen would be a bit tiresome. The need for panning is greatly reduced with the 80 column screen but on the 40 column PET, for which WordCraft was originally written the pannning facility can greatly reduce the strain on thinking out layouts.

WordCraft is orientated around pages and chapters. This is ideal for long documents and books but for many applications this is too great a depth. The top of the screen is packed with information but I felt that it was mostly useless. You are given the documents name, the date it was created, the chapter file name and the chapter number. On the next line you get the cursor position, giving both the line and column numbers. Next to this is the page number and the number of pages in memory. The number of characters that are free and the disk drive that the chapter came from are the last items of information which are given. The next two lines are taken up with command and control mode inputs and error messages. The next line is used to give a visual indication of margin and tab settings which is probably the most useful of the five lines. I personally think that the loss of 5 lines from a screen with 25 lines is too much. Really most of the information should only be put onto the screen when it is required by the user and not be there all the time wasting space.

The program has a powerful search and replace facility which can be used both to look through the text for specific strings of characters and also be used to search and replace where necessary.

There is a function which can be accessed to set up the fifth line of the screen. This function is called Ruler. It allows the margins to be set by simply positioning the cursor at the desired margin setting and then pressing the 'greater than' key. Tabs are set and cleared in a similar manner. As an example of this functions power I will give a small demonstration:-

> A margin can be set to any position and will remain at that setting until another setting is given. The margin setting used here is 10,40.

This has reset the margin width to 5,45. When moving through the text the margin settings move according to the embedded commands, a very powerful and well thought out feature.

Getting and putting files onto disk can be a fairly harrowing experience until the sytem is worked out. It took me about five attempts to get this text from disk but after a while it becomes easy. The difficulty arises from the fact that there are a great many options open to the user for controlling where the text is going to be put and what it is going to be called. To provide simple backup facilities the following command replaces the old version of text on disk with the latest version in memory:-

s,,,,r

WordCraft does not make provision for scratching unwanted files. This is left for the user to do in immediate mode from BASIC. This is a bit suprising considering the depth to which the rest of the disk commands are controlled.

WordCraft has a very powerful form letter capability. As a test for this part of the program a simple form letter was written to try it out. I wrote the letter and at every point where I wanted information inserted I entered control mode and pressed ?. WordCraft then asks for a letter to be input, as soon as this has been done a tick and the letter appear on the screen in reverse field. This is a very powerful feature as it gives the capability of using the same bit of information in different parts of the letter. When I had finished the letter I saved it on disk. I then wrote what WordCraft calls a fill-file which consisted of names and addresses and saved this on disk. When I recalled the letter and gave the fill command the information was taken from the fill-file and displayed on the screen while being printed. The letters were printed as fast as the Spinwriter could handle them and I was very impressed by the ease with which it was all accomplished.

Comparison between WordCraft 80 and WordPro 4

These are the only two serious word processors available at present for the PET. Both are very powerful and have very similar functions but they tackle the various problems presented by word processing in different ways.

The best feature of WordCraft is its immediate formatting which all the professional, dedicated word processing systems have. It is this feature which gives the greatest differences between the two systems. When the margins are set with WordCraft the user types in text and when the right hand margin is exceeded the word is moved to the beginning of the next line. An obvious advantage of this is that the user does not have to take special care when editing with words which are split at the edge of the screen, as with WordPro.

WordPro only uses two lines of the screen to display information as opposed to WordCrafts use of five lines, it makes the screen seem far less cluttered and because the lines used by WordPro are not in reverse field the information is far less distracting.

WordCraft has a very comprehensive directory which may be useful in some circumstances but WordPro has the ability to call up documents from disk by using the cursor to point to the name of the file. When the backslash is pressed the file is called up. As I pointed out in the review of WordPro the disk directory replaces the current memory content. This is annoying as BASIC4 has a command called CATALOG which prints the directory to the screen without having to load it into memory.

WordPro has a more powerful disk handling capability, a good example of which is to press the STOP key and then '1' to initialise drive number 1. To achieve this with WordCraft means entering command mode then typing 'u,i,1' seven key strokes as opposed to two. I never had the time to examine WordCrafts filing capability deeply but it looks very powerful, with everything being stored as chapters of a document, this removed the need for the operator to have to worry about what page comes next. WordCraft lacks many of the disk handling capabilites needed for succesful housekeeping, it seems incredible that you have to go into BASIC to scratch files.

I found the embedded commands particulary difficult to master on WordCraft, especially when it came to delete them, a command is shown in the text by a reversed field. To display the commands held on a line RVS 'c' is pressed, some of the embedded commands appear on the screen as one character and have to be deleted by using another key, though there are only two cases where this occurs.

WordPro has what is called Insert mode. This is useful for editing text and making insertions. Text is input by the user and rather than deleting what is underneath the cursor it moves the text along as charcters are typed. WordCraft has a rather primitive system where by RVS 'inst' moves the text along by about 8Ø characters. Text is then input and when finished RVO is pressed and the unused space is deleted.

The two programs store text in different manners. WordCraft stores text sequentially and packs as much into memory as possible. WordPro stores text in 8Ø character lumps so if there is only 1 charcter on a line then 8Ø characters will have been lost from memory, this gives WordCraft a very large advantage in the amount of text which can be held in memory. When panning up and down through the text with WordPro it is actually memory which is being examined. This is why WordPro has the ability to pan through text very quickly. WordCraft has to format the text as it appears.

The standard letter capability of WordCraft is superior to that of WordPro in two respects. The fill-file is easier to write and the information is taken from disk rather than having to be resident in memory thus allowing far larger address files. Secondly, if, for example, you wish to personalise a letter by having the recipients first name mentioned a few times, with WordCraft this is simple. All that needs to be done is to input the same identifying character throughout the letter and when it is printed the name will be taken from the fill-file and inserted at the different locations. With WordPro the name would have to be typed in for each location in the letter.

Which of the two would I buy? Both have ardent followers who proclaim the advantages of their system. To me WordPro is more user friendly but that may be because I have used WordPro ever since I joined Commodore. However, the command system with WordPro is easier to access using where possible only one or two keys. The main decision which the user will have to make is whether the text is to be formatted as it is input or if you are willing to allow it to be typed in with words being split on the right hand side of the screen and then only formatted when it is output.

KRAM Barry Miles

This package is a very interesting one, in that it successfully provides an alternative to the Commodore Random Access system, which is based on an advanced variation of the Indexed Sequential Access Method and is similar in action to the Relative Record system now available as part of BASIC 4 but with advantages. It enables a user to program random access and sequential access to files in a single operation, because although data is stored in random order, the system permits access to records in ASCII sequential order and another facility will retrieve them in the reverse order.

The search method used is efficient, so that data can be retrieved in between one and two seconds typically, with a worst case of about 2.5 seconds.

The user designs the file structure himself and the KRAM system is really a "Do It Yourself Database Kit", although a Mailing List program is included in the package, to act as a demonstration of the system and also containing subroutines for inclusion in one's own program. The system is operated by means of a number of machine code routines which are accessed by SYS commands, but rather cleverly, the commands have been designed to cause a degree of documentation, in that a key string must be set up before the SYS command is given and that key string must contain the KRAM keyword covering the action being carried out. e.g. If a database is being set up, the keystring must contain the word "Create"; if a record is being added, the keystring must contain "Add" and so on. Later versions will add these commands as additional BASIC commands, thus eliminatung the need for the SYS commands completely.

Some care is needed at the planning stage to use the system most effectively, but the 40 page manual gives a great deal of help. A major element in the efficient use of the system is the ability to have up to five KRAM files open at once.

This enables you to use fields within a record to access other records in another file; thus avoiding waste of space by the unnecessary duplication of data in a number of records, or in exteme cases, in every record.

In order to give the system an adequate trial, I used it as an opportunity to set up a program for a problem which had been a headache for a considerable time. My disks were getting more and more full and I was in danger of never being able to find a particular program. The situation was becoming more acute, as wordprocessing and VisiCalc datafiles began to proliferate.

What was needed was a directory of directories, with the ability to access fROM the keyboard, data about any program, including the ID and name of the disk on which the required file was to be found. I also wanted to print out a catalogue of programs, sorted by category and in alphabetical order within categories. Using conventional sorting procedures was a problem because of the limitations of RAM space and of the inability to sort a disk full of data in one go.

Thus I had the opportunity to give KRAM quite a going over. The method used was to use WordPro to change a directory, by adding as a prefix to the filename the category of the file eg. "U" for utility, "G" for game, the number of the disk,(3 characters) and a quality code, which also identified sequential, WordPro & VisiCalc files, (one character). It would have been equally easy to use arrays to store the subsidiary data, but I wanted to test out the use of various KRAM files simultaneously.

The data was put into a continuous string of 21 characters and stored in a sequential file, pending its insertion into the catalogue, using KRAM, with the whole of the string acting as a Key. Any record could now be retrieved instantly, updated, or deleted. A formatter section of the program then printed out a categorised list, with the various codes indicated above translated into their significance.

The system works extremely well and various handy KRAM facilities emerged. If you cannot remember the full key for a record, you can use a partial key and then browse through the records in ASCII sequence until you find the one needed. If you attempt to add records which are duplicates into the KRAM file, the information is not duplicated. This was particularly useful in my application because a new version of the sequential file of the data could be produced, updated by the addition of extra programs added to the disk and the data run through the KRAM input routine, knowing that KRAM would reject all but new records.

The package comes a a disk-based program together with the inevitable security ROM. It is not cheap, but if you can make use of its facilities, it represents good value for money and Calco Software, its U.K. distributors know all there is to know about the package and support it fully, even to the extent of sending out a newsletter to customers!

If you need even more sophisticated routines, then SuperKRAM is on its way and pending a reviw of that, let me just mention multiple keys and limitless numbers of files open.

Superchip Barry Miles

This ROM is advertised as a supplement to the Toolkit, with which it is fully compatible. It contains a large number of routines which should give something for everyone. For instance REPEAT, which can be set to your own choice of delay, speed of repeating, is a very pleasant aid. Some will question the value of the choices given but to my mind it can be aggravating if the repeat delay is too long, or the repeat frequency is too slow or too fast. It is worth remembering that the average reaction time is 7/1Øths of a second and racing drivers react in 3/1Øths. Slow for some is fast for others! Similar comments apply to the ability to type most of the common Basic keywords, by holding down The RUN/STOP key,(redesignated control by Supechip) and hitting one other key, usually the obvious one: e.g."R" for Return. Some have said this is pointless, since the use of the initial letter and shifted second letter will produce the same result. I cannot agree with this however, because until the line is listed, you cannot tell whether the 8Ø character limit has been exceeded, nor can you read the code easily. At times it is helpful to exceed 8Ø characters, to produce compact programs and to make alteration of existing programs harder but normally clarity is much more helpful. The Superchip way is usually more satisfactory.

A RETRACE facility is provided which has considerable use in debugging. With this enabled, you may discover the last 1Ø lines executed before you broke into the program. The break will be caused by use of either a bracket key which Superchip redesignates "STOP", or by use of another facility:"HOLD", which stops everything, including the clock, until Return is pressed.

Screen-handling has been considerably enhanced, so as to provide many of the facilities of the 8Ø32, some of them more readily accessible than on that machine! Erase begin, erase end, delete line, insert line, scroll up, scroll down and even a window of up to 9 lines are all available. All these functions are available both from the keyboard and from within programs by way of SYS commands.

The greater ease of program editing provided by these facilities is likely to be much appreciated. For instance, having listed a number of lines, it is very pleasant to be able to create a line between two others and see exactly what it will look like, especially if you are using indentation to assist clarity.

ESCAPE will enable you to jump out of programmed cursor mode at will, using the obvious command "control quote ("). This prevents a large amount of frustration when seeking to modify strings. You can also toggle between graphics and lower case mode at will. Convenient if you cannot remember 59468, or whether "12" or "14" means lower case !

SHRINK is a very useful command, since it calls up a routine which immediately removes all remarks and unnecessary spaces from your program. This will enable you to save memory and speed up execution of most programs, in some cases dramatically. Some care must be exercised because if you have REMARKS which are the target of other statements (e.g. GOTO), these will be deleted. However, the routine is a considerable aid and it is not even necessary to store two versions of your program, one complete and one shrunk. The routine is so fast, that it is more convenient and cheaper, to shrink each program just before running it.

REVERSE used in a program will change all normal characters into reverse field and vice versa. The area of effect can be defined.

MOVIT will copy any area of ROM or RAM into any RAM area. (Although it does not relocate machine code, merely moves it.) If you wish to store screen displays for later recall, this does it for you.

It is possible to store a user-definable message in the first cassette buffer for display by direct call, or by means of a programmed SYS command.

Finally you may call up to 1Ø user-defined machine language routines, provided that you are not using the window scrolling routine.

The manual is very informative and the package is demonstrated on a disk or tape which will show you everything.

Very good value!

Incidentally, Toolkit for Basic 4, both 4Ø column and 8Ø column machines is now in the country and Supersoft tell me they had it first!

PETPACKS

Pete Gerrard

As promised earlier, a list of the new titles in the cassette library for this winter. Before going onto the new releases, there have been a number of changes to the existing catalogue, designed to improve the overall quality of the programs available from Commodore. A lot of the old titles have disappeared, to be replaced by an excellent set of new ones, to give you a truly superb selection of programs for your Commodore PET.

The following 20 titles are no longer available from Commodore :-

MP002, MP004, MP009, MP016, MP022, MP023, MP024, MP025, MP027, MP033, MP049, MP050, MP054, MP056, MP057, MP070, MP071, MP077, MP080 and MP081.

This is a rather drastic start to a review of the new releases in the cassette library! But rest assured, these new titles are better than ever! There are in fact thirteen releases this time around and they are as follows :-

Order No.	Title	Retail Price inc. VAT
MP090	Simulator #1	10.00
MP091	Physics Pack #5	10.00
MP095	Adaptive Kybd Trainer	15.00
MP096	BASIC Aid	10.00
MP097	Extramon	10.00
MP098	Lib. of Sub-Routines	7.00
MP099	Labyrinth	7.00
MP100	Laser Tanks	7.00
MP101	Southern Hangman	10.00
MP102	Northern Hangman	10.00
MP103	Life	7.00
MP104	Cosmic Jailbreak	7.00
MP105	Crazy Balloon	7.00

Also, we have new versions of two of the old programs in the catalogue, namely Linear Circuit Analysis (MP047) and Treasure Trove of Games #8 (MP052), which now consists of real time versions of Sorcerer's Castle and Jason and the Argonauts complete with sound effects (provided you have a sound output of course!)

Although descriptions of each of these programs can be found in the enclosed PET Pack and General Disk Library Catalogue, I'd like to say a little bit more about some of the programs.

Arcade Games

Laser Tanks, Crazy Balloon and Cosmic Jailbreak are the next three games in our Arcade series and carry on the grand tradition started by Invaders some 9 months ago. Unlike those earlier arcade games, these are home-grown products and are every bit the equal of their mainly Japanese counterparts. Although it's difficult to say which one of these is the best of the three, it's Cosmic Jailbreak which has proved the most popular since its arrival at Commodore a short while ago.

You are in command of a space-ship, which can move across the bottom of the screen and which can fire lasers at the enemy above. You have three bases to hide behind, but it doesn't take long for these to be wiped out, both by yourself in a mad panic to get at the aliens and by the aliens as they aim at you. At the start of the game, the aliens are displayed in two vertical lines at either side of the screen and gradually they all start walking inward towards the centre. In the centre is a 'jail', made up of '#' characters, which is holding three aliens prisoner. The aliens are trying to free their trapped compatriots and your job is to stop them, shooting them down as they walk to and fro, gradually picking up and taking away pieces of the jail, getting ever nearer to their companions. All the time they're firing bombs at you and if one of them hits you, you've had it! Initially you start out with three 'lives', gaining a bonus life when you score 3000 points and another one every further 3000 points. There are two ways in which you can lose a life - one is by getting hit by a bomb, as we've said and the other is if the aliens actually manage to free one of their allies. From time to time a mystery space-ship will fly across the top of the screen and you can gain a random score by shooting this down - useful when you need a few more points to get an extra life.

If you look at the picture of Cosmic Jailbreak in the catalogue, you'll see a good example of some of what we've been talking about. The mystery space-ship is flying across the top, there's an alien on the left, walking back with a bit of the 'jail', an alien on the right has dragged one of his companions out of the jail and at the bottom of the screen you can see the display of your ship when it's losing a life.

One of the most appealing aspects of the game is the way it subtly becomes more and more difficult as you clear each successive screen of aliens and progress onto the next level. The level number is indicated in the top right hand corner of the screen and also on the bases - again, you can see this if you look at the picture in the catalogue. Occasionally you will get what is known as a 'runner' - this is an alien on a kamikaze mission, when all the others stop moving and he races back and forth, grabbing bits of jail until he gets his compatriot away to the side of the screen. You've got to shoot him, because he won't stop until

you do. On level one, all the other aliens stop when a runner appears. On level two, they still stop moving, but will carry on firing at you. On level three, an additional 'alien' appears, but not in line with all the others. This one can appear at any time (usually an inconvenient one!) and will appear immediately above your bases and will follow you across the screen, firing at you until you get him. On level four everything gets a bit more frantic. Level five and another set of aliens appear, this time six little ones, which also appear above your bases and run across the screen from left to right, firing at you all the while. If you manage to shoot all six of them, they simply reappear to make life that much more difficult. And so on to level nine - no other 'nasties' appear, it all just gets faster and more furious. After level nine (and incidentally, I've only seen somebody do this once!) you get a bonus of 5ØØØ points and after the game starts again you're back at level six, only this time it's faster than ever.
A truly great game and possibly the most addictive one we've ever released. Write in with your highest scores and we'll keep track of them as the newsletters go by. The highest score we've got on Cosmic Jailbreak is 49,98Ø and that took about 4Ø minutes to do. If you get a higher score, write to me at the address at the end of this article and we'll keep publishing the highest scores in the newsletter. Incidentally, you can write in with your highest scores for all the other arcade games as well and we'll form a national league of games lovers. If somebody writes in with an amazingly high score, they'll be asked to prove it!

Basic Aid

Of the other releases, two of the most interesting are BASIC Aid and Adaptive Keyboard Trainer. Taking them in that order, BASIC Aid costs just ten pounds and is a set of programs that add an impressive number of extra commands (over 2Ø) to PET's BASIC. Note that this will only work on BASICs 1 and 2 at the moment, although work is well in hand for a BASIC 4 version. The commands include such useful features as renumbering, block deleting, trace, auto line numbering, find, change, repeat, all the dos-support commands (plus a few extra!), merge and so on. The program occupies just 2 or 3K of your PETs memory (there are different versions, depending on whether you're using a disk drive or not) and can be relocated anywhere you like, so that it can work on any of 8, 16 and 32K PETs. Complete with instruction manual, this is an invaluable aid to any programmer. What else can I say - we use it all the time!

Adaptive Keyboard Trainer

The Adaptive Keyboard Trainer breaks new ground for Commodore. It is a touch typing course and can be used by people of all ages to teach themselves to type, on the PET keyboard and to type with all the speed and proficiency of a professional touch typist. The program is exceptionally well-written and comes complete with a manual, explaining a little bit more about how the program works. I've used it and my typing speed has increased considerably - it has to for articles as long as this!

On running the program, you're asked for a choice of alphabetic or numeric typing practise. Typing 'A' for alphabetic will then clear the screen and ask for a difficulty level (a number between Ø and 9). If this is the first time you've used the program, you should type in 'Ø' and then you'll see a representation of the PET keyboard on the screen, with a symbol over one of the letters, indicating that this is the one to type. This symbol moves along as you type, jumping from letter to letter. Beneath the keyboard is a display of the letters you're being asked to type and ideally you should only look at those letters and not the keyboard in front of you or the one on the screen - it's the only way to learn to touch-type. If you'd selected numeric mode, everything would be the same, except you'd see a display of the PET numeric keyboard rather than the alphabetic one.

This is where the program comes into its own. As you're typing away, the program continually monitors your performance and only sends you onto the next difficulty level if your skill at the particular level you are on is good enough to warrant it. If it isn't, you stay on the same level. If you do really badly, you get dropped down a level until your skill improves again. Thus the program is constantly keeping an eye on your progress and responds accordingly.

This is the sort of program that could find many uses - programmers who are tired of spending hours typing listings in; people who simply want to speed up their typing; in schools for kids using PETs; as a professional touch-typing course, with some kind of multi-PET system and many other uses besides.

There are other courses planned, as the programs extend towards the 8Ø32 graphics keyboards, the new VIC and so on. If you thought you'd never get used to all the different keyboards Commodore is producing these days, this and the other programs to follow, will provide the solution. All details will of course be published in the newsletter.

Before carrying on to the competition, I'd like to say something about the compatibility of various programs between the different types of PETs. The introduction of BASIC 4 meant that a number of our programs and in particular the arcade series, had to be modified to run on these new 4Ø column PETs and I'm happy to say that all of the required changes have been made. Consequently, from receiving this newsletter, when all of the new releases will be available, all cassette products will work on BASIC 2 and 4, (with the couple of exceptions noted in the enclosed catalogue). Where a program (like Invaders for instance) requires three different versions to run on all the PETs, we are just supporting the two most current BASICs, namely 2 and 4.

In order to forestall a number of telephone calls, we are currently NOT planning to release any cassette products for the 8Ø32, but instead are concentrating on producing the best available material for the 4Ø column PETs.

And so on to the competition. First of all, thanks to everybody who has sent in a program so far. I seem to have sparked of a lot of creative genius! Since this has proved so popular, I'm going to extend the closing date until we go to press on the next issue. This gives any of you who haven't sent anything in so far a last chance - you've got about three weeks to get those programs in! I'll detail all the winners in the next issue.

Send your arcade high scores and any programs you've written, to :-

PETe Gerrard,
Comoodore Business Machines,
818 Leigh Road,
Trading Estate,
Slough,
Berks.

Guten Topps der Morninks

Ich haffing readen der Brief in die PET doggies magazinens von Folume Drei (nummer eins) entitled "Achtung Alles Lookenpeepers".

Der importantness muss stressed be. Die instructions und Befehlungsweisen in diesen Brief to be taken mit seriousness must. I, of the Englisch speaking perfect am, muss zu meine Kameraden in ze Computers Game einen Warnung given. So bitte to be der close attention giving!

Das "PET doggies good-boy pat-pat" Maschinen is wie gesagt einen Complizierten instrumenten der high Technik. Aber das BASIC der food of the "PET doggies good-boy pat-pat No! it's not walkies times" ist relatively (Ach! - der Einstein everywhere gets!) simpel. Aber (und hier der Warnung fuer Beginning peoples ist) nicht so easy der Filing handlings ist.

I therefore already hoping am that all peoples not mit der Fingersgepoken but especially already that the peoples of the clever knowing how to do it types in respecten of the "PET doggies good-boy pat-pat No! it's not walkies time - sit! play dead!" maschinen their knowledge pass on will in understanding easily articles.

Mit Spitzensparken Blinkenkights

Ich bleibe Hochachlungs voll

Herr K one over the eight.

P.S. Dr. Who has copyright on K9!

Save yourself £23.00!!!!

A common equipment failure noted on Training Courses is caused by the PET-BIRO interaction.

During the high fervour encountered on a course the biro wanders, seeking the calm and sanity found only in the dark recesses underneath the PET.

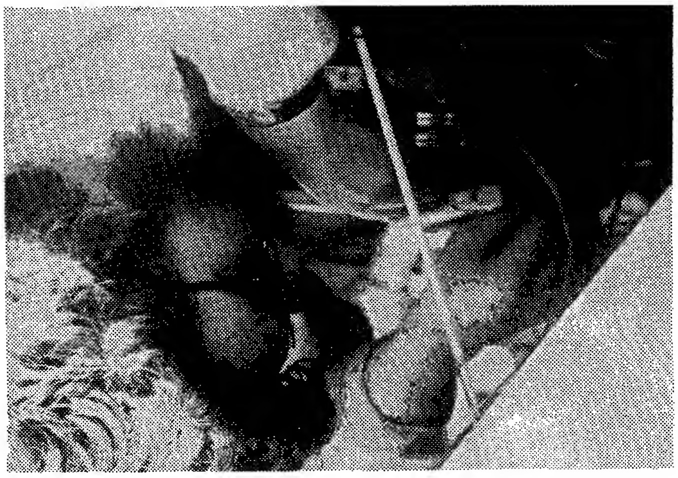
The operator, now reaching the peek of his excitement, locates this errant pen and, in order to retrieve it, lifts the PET up and encounters nasty cracking sounds as the PET-IEEE cable connector attempts bravely to withstand the force.

The result of this scenario is one recovered Biro and one smashed and useless connector.

The remedy for this ailment is simple ..... tip the PET on its side instead of its back!!!!!!!!!

Lo and behold you have just saved yourself £23.ØØ.

C. Palmer
Training Department



John Collins inside a PET.

PET Makes Prescriptions

An Ipswich doctor has been prescribing his practice with a PET micro computer system.

The system has been introduced to provide, what seems like, a foolproof method for dealing with patients' requests for repeat prescriptions.

Previously when patients rang in wanting 'repeats' they would be able to provide the receptionist with little more than the colour and shape of their drugs. This meant that the receptionist had to search through files, writing out their prescriptions in often illegible handwriting, for the doctor to sign and then sending it off to the patient.

It was also difficult to for a check to be kept on the amount of drugs that had already been supplied to a patient. However with the doctor's new computer system the patient simply keeps a record card with a code number and the drugs that are being taken written on it. When they want a new prescription it is sent in to the receptionist, the code number and drugs required are fed into the machine and a neatly typed prescription is produced.

The machine also prints out and stores information relating to the amount of drugs and the date on which they were prescribed.

Another advantage of the machine is that at the press of a button the doctor can find out how many people are on a certain drug. This means that if some information should suddenly come through warning that a certain drug could be dangerous all the people on that drug can be quickly tracked down.

So far the doctor, who can not be named due to the strict rules laid down by the Genneral Medical Council says that his PET has met with great enthusisam from both colleagues and his recepionists.

The system has been on display at the Royal College of Physicians in London and he is hopeful that GP's across the country will adopt it.

The doctors story is all the more remarkable because he is not a qualified computer expert. He bought books on programming and taught himself how to use a computer which he bought at a shop.

"Using a computer to record information on patients and diseases is something I had been thinking about for many years", the doctor explained. "But I could not do any more about it because of the cost. When a computer came down within my price range I went straight out and bought one."

For more details about the package contact Patrick Dixon at Medicom, phone 01 579 5845.

The picture shows Mrs Lenna Astridge, a receptionist at the Ipswich surgery operating the system.

TEACH-IN

Mike Gross-Niklaus

1. More on program formatting!

Regular readers of this column and the PRINTOUT colomn which preceded it will have noticed that I put great emphasis on a neat and organised layout for programs. The reasons for doing so are well summed up in a most useful book from Addison-Wesley written by John Nevinson and called "The little book of BASIC style". Not that the book is little, containing as it does 150 pages of valuable suggestions, supported by plentiful diagrams and cartoons, on how to layout your programs.

The main point that he makes is that programs should be set out in much the same way as books or reports, with plenty of tabulation, blank lines and headings. If you were writing a report to be read by others, you would I am sure, break it down into numbered or lettered sections. Mr Nevison suggests that time spent in formatting your programs in a similar way saves much more time in the debugging stage of program development.

My experience is that the effort spent on a typical Basic application breaks down as follows:-

Specification	30 - 50%
Coding	20%
Debugging	50 - 30%

Obviously anything that can be done to cut down the debugging effort is worth-while. I am absolutely certain that neatly formatted programs listed on a printer, with plenty of space between the lines, will help you significantly in reducing the time necessary to produce a robust working program.

The subjects in this issue of TEACH IN are discussed with that concept in mind.

2. Multi Digit Add (Assembly code program)

The Training Department have now run their first two very successful Assembly Code courses, with more on the way in the new year. The aim of the course is to get people to the point where they can create useful programs in machine code, using the full Commodore Assembler package or the Mini Assembler in that most useful utility, Extramon, which forms part of the Assembler package. Extramon is a slightly more advanced version of SUPERMON which is published and upgraded in CPUCN.

The problem with assembly programming is that once you have mastered the tools of the trade, and look around for applications, most of them appear to involve external devices with their own complexities of coding, handshaking and timing protocol. I recommend to students that initialy they look around for simpler routines which they can add to BASIC programs.

One or two examples written by students on the courses included:-

> Sounding a tone blip, different for every key, each time a key is pressed. This could have some value for blind users. One student extended this to scanning each line on the screen and producing a different sound pattern for each character present. (Roll on, the Pet speech synthesisers).
>
> Flashing the screen into reverse and back. This is an exercise done by every student in about the fourth hour of the course.
>
> Moving a large pattern about on the screen much faster than in BASIC. One or two students extended this to moving the pattern over the screen contents, ie, once the pattern moved on, the original screen contents reappeared. One application might be in games animation: another in planning, where you might want to lay one diagram on top of another and later 'peel' it off again.
>
> Searching a binary array created in a protected area above the top of BASIC. One application for this is the creation of your own Block Availability Map for Direct Access operations using DOS1. Another, which I used in a program analysing the PET show survey, found matches for a particular pattern of answers. E.g. "What percentage of replies put 'YES' for questions number 1,5,13,22,41,63 and 70 and 'NO' for questions 4, 12, 31 and 64"? In BASIC this takes ages, but using a short machine code routine, called by SYS, the search time seemed instantanious.

As an example of what can be done I've taken a suggestion from the OSBOURNE/McGRAW-HILL book written by Donahue and Enger and entitled "PET/CBM PERSONAL COMPUTER GUIDE". In section 5, on page 190 onwards, they describe a way of performing arithmetic, using strings, which gives as many digits accuracy you need, up to 255 places, as opposed to Microsoft BASIC's nine digit accuracy. Those of you who have tried finding out what 2 to the power 235 is or what factorial 90 evaluates to will understand the problem. Some business applications require 12 digit accuracy.

The method involves holding the numbers

to be added as strings and performing the addition character by character, starting at the right-hand end. In practise this turns out to be noticably slow. It seemed a good candidate for turning into a machine code routine.

The method I used was to set up A$,B$ and C$ as the first three variables in the variables table. That way, by looking at the start of variable pointers in locations 42 and 43, the routine can find the strings. Once the values to be added are assigned to A$ and B$, with the decimal points aligned in the same position counting from the right, the addition is performed as in the BASIC version. The program has to take account of any carry over from the previous digit, skip over the decimal point and adjust the result because ASCII values for the digits rather than absolute values are held in the strings. E.g. "123" is held as values 49 50 and 51 in successive memory locations. The result is placed in C$. A listing for the BASIC program is shown below.

```
.:  033C 03 04 0A 0B 11 12 00 00
.:  0344 00 00 00 00 00 00 00 00
.:  034C A2 09 B5 00 9D 42 03 CA
.:  0354 10 F8 A5 2A 85 00 A5 2B
.:  035C 85 01 A2 05 BC 3C 03 B1
.:  0364 00 95 02 CA 10 F6 A2 00
.:  036C 8E 3A 03 A0 02 B1 00 38
.:  0374 E9 01 8D 3B 03 AC 3B 03
.:  037C B1 02 C9 2E F0 19 18 71
.:  0384 04 6D 3A 03 A2 00 8E 3A
.:  038C 03 38 E9 30 C9 3A 30 07
.:  0394 E9 0A A2 01 8E 3A 03 91
.:  039C 06 88 10 DC A2 09 BD 42
.:  03A4 03 95 00 CA 10 F8 60 00
.:  03AC 00 00 00 00 00 00 00 08
.?
```

The routine was developed using the full Commodore Assembler and once again proper formatting of the program helped the debugging process enormously. The source coding is shown below and I think that even those of you who are not too familiar with assembly code will still be able to follow most of the program through.

```
READY.

1000          ;M/C ASCII STRING ADD, MULTI DIGIT
1010          ;
1020          ;MIKE GROSS NIKLAUS
1030          ;
1040          ;5/8/80
1050          ;
1060          ;==================
1070 POINT0 = $00               ;POINTS AT THE START OF BASIC VARAIBLES
1080 POINTA = $02               ;POINTS AT A$
1090 POINTB = $04               ;POINTS AT B$
1100 POINTC = $06               ;POINTS AT C$
1110 CARBYT = $033A             ;ACTS AS CARRY REGISTER
1120 LENGTH = $033B             ;LENGTH OF STRINGS
1130 ASCPNT = 46                ;DECIMAL POINT ASCII VALUE
1140          *=$033C
1150 ZERPNT .BYTE 3,4,10,11,17,18
1160          ;OFFSETS TO FIND A$,B$,C$ IN BASIC VARIABLE TABLE
1170 TEMZER .BYTE 0,0,0,0,0,0,0,0,0,0
1180          ;==================
1190 SAVZER LDX #9              ;SAVE 1ST 10 ZERO PAGE LOCATIONS
1200 NXTZER LDA POINT0,X
1210        STA TEMZER,X
1220        DEX
1230        BPL NXTZER
1240 ;
1250 SETPNT LDA $002A
1260        STA POINT0          ;SET UP POINTER TO 1ST STRING
1270        LDA $002B           ;$2A, 2B IS THE POINTER TO THE START OF STRINGS
1280        STA POINT0+1
1290        ;
1300 SETLOC LDX #5              ;INSTALL POINTERS TO A$, B$, C$
1310 NXTLOC LDY ZERPNT,X        ;USING PRESET OFFSETS TO FIND THEM
1320        LDA (POINT0),Y      ;IN THE BASIC VARIABLE TABLE
1330        STA POINTA,X
1340        DEX
1350        BPL NXTLOC
1360 ;
1370 INITCB LDX #0
1380        STX CARBYT          ;SET CARRY TO ZERO
1390        ;
1400 MOVLEN LDY #2              ;USE LENGTH OF A$ AS DEFINED BY 3RD BYTE
1410        LDA (POINT0),Y      ;IN BASIC VARIABLE TABLE TO SPECIFY SAME
1420        SEC                 ;LENGTH FOR ALL THREE STRINGS
1430        SBC #1
```

```
1440         STA LENGTH
1450 ;
1460 ADDSTR LDY LENGTH       ;SET UP TO DEAL WITH ALL CHRS IN A$ & B$
1470 NXTBYT LDA (POINTA),Y   ;IS THIS CHR OF A$ A DECIMAL POINT
1480         CMP #ASCPNT
1490         BEQ STORES      ;YES, SO ARITHMETIC JUST COPY TO C$
1500         CLC             ;NO, SO PREPARE TO ADD
1510         ADC (POINTB),Y  ;ADD CHR FROM B$
1520         ADC CARBYT      ;ADD CARRY IF ANY
1530 ;
1540 ZERCYB LDX #0           ;RESET CARRY TO ZERO
1550         STX CARBYT
1560 ;
1570 ADJASC SEC             ;ADJUST SUM BECAUSE TWO ASCII CODES WERE ADDED
1580         SBC #$30
1590 ;
1600 OVRTEN CMP #$3A         ;IS THE RESULT > ASCII 9?
1610         BMI STORES      ;NO! SO STORE THE RESULT
1620         SBC #$0A        ;YES, SO ADJUST TO REMAINDER PLUS
1630         LDX #1
1640         STX CARBYT      ;CARRY
1650 ;
1660 STORES STA (POINTC),Y   ;WHATEVER HAPPENED, PUT THE RESULT IN C$
1670 ;
1680 MOVEON DEY             ;NEXT CHR UNLESS
1690         BPL NXTBYT      ;ALL ARE DONE IN WHICH CASE....
1700 ;
1710 RESZER LDX #9           ;LEAVE ZERO PAGE AS YOU FOUND IT
1720 NXTRES LDA TEMZER,X     ;BY COPYING THE 10 BYTES STORED BACK AGAIN!
1730         STA POINT0,X
1740         DEX
1750         BPL NXTRES
1760 ;
1770 ALLOVR RTS             ;BACK TO BASIC
1780 .END

READY.
```

```
10 REM TITLE: MULTI DIGIT M/C ADD
20 REM BY    : MIKE GROSS NIKLAUS
30 REM DATE: 13/10/80
40 REM FOR   : TEACH IN
99 REM"

2000 REM A$,B$,C$ FIRST IN VAR TABLE
2010 A$="0": B$="0": C$="0"
2099 REM"

3000 REM WHAT ACCURACY IS REQUIRED?
3010 INPUT"HOW MANY DIGITS";D
3099 REM"

4000 REM CREATE A$,B$,C$ THAT LENGTH
4010 FOR I=1 TO D-1
4020 A$=A$+"0": B$=B$+"0": C$=C$+"0"
4030 D$=D$+"0": NEXT I
4099 REM"

5000 REM GET THE NUMBER TO BE ADDED
5010 INPUT"1ST NUMBER PLEASE";N1$
5020 INPUT"2ST NUMBER PLEASE";N2$
5099 REM"

6000 REM ADJUST TO LENGTH
6010 A$=LEFT$(D$,D-LEN(N1$))+N1$
6020 B$=LEFT$(D$,D-LEN(N2$))+N2$
6099 REM"

7000 REM DO M/C ADD
7010 SYS 844
7099 REM"

8000 REM DISPLAY RESULT
8010 PRINT"THE RESULT IS:-";C$
8099 REM"

READY.
```

The final listing is a hex memory dump of the routine for those of you who want to enter the program using the monitor or BASIC DATA statements.

3. Vertical tabulations in listings.

Many of you will know how to indent lines of BASIC, to highlight nested FOR..NEXT loops for example. One way involves putting a colon followed by spaces at the start of the indented line. Another, invisible, method requires you to type any shifted character followed by spaces at the start of the indented line. When listed the shifted character disappears but the remaining indentation is unaffected.

It's also possible to obtain vertical tabulation, ie. blank lines on listings both to the screen and to the printer. Although the method may seem rather obscure, it saves quite a bit of memory that would have been used to hold characters highlighting the 'start of block' REM statements. I've been using a development of the original idea as published in the "Liverpool Software Gazette".

The trick is to place a CHR$(141) followed by two CHR$(10)'s at the point in the program where you want a blank line to occur. (Ideally a CHR$(13) would do the trick by itself, but apart from poking, I can't find a way of typing it into the program). The best place to put these characters is in a REM statement at the end of each program 'Block'. The way to do it is to type the line number, say 2099, then REM then two quote marks. Delete one of the quote marks, press REV, then shifted M, which will place a CHR$(141) in the program line. Finally, two unshifted Js will create the required CHR$(10)s. The original idea, in the LSG, suggested just CHR$(141) which is the shifted return character. However, the Commodore printer needs the CHR$(10)s, both of them, as well.

The listing of the BASIC part of MULTI DIGIT ADD includes a number of these REM vertical tabs in REM statements at 99, 2099, 3099, 4099, 5099, 6099, 7099 and 8099. To put them in, I typed in line 0099 REM etc, found all the header REM statements using the FIND facility in BASIC AID, then repeatedly edited the line number and pressed return. BASIC AID? you ask? More of that next time!

BUSINESS USERS COLUMN

Barry Miles

Some readers will not yet have purchased their machinery or having done so, will not yet have installed it in their business. Therefore some general observations about the pre-installation, installation and implementation stages will not be out of place.

It is worth remembering when considering these matters that computers are not everyone's idea of fun! As an enthusiast myself, I must admit that when there is a lot of work to be done on the machine, I dislike it as much as the next person!

What a business needs if microcomputerisation is to be a success, is enthusiasm for the innovation and a willingness to accept a period of teething troubles, with a belief in the success of the venture in the long run. It is apparent therefore that at least as much thoughtful consideration should be given to encouraging an environment to develop in which the installation of the system is welcomed as a beneficial innovation which will help the people concerned to maintain and improve their achievements at their jobs. It really comes down to a matter of motivation and conviction. These do not come easily if one simply presents staff with a fait-accompli. Many a new installation is likely to fail if the staff come in one morning and are told "We've got a microcomputer now and you're going to make it solve all our problems".

Most people placed in this situation will rebel and some may even promise themselves that it will never work.

The only safe way is to involve the people who are going to use the system in the decision-making process. Take those who are going to use the machinery to the computer shows; let them try various programs out for themselves and listen carefully to their opinions.

The current state of microcomputing is such that you really need at least one enthusiast in your office, who will get to know all the idiosyncracies of the equipment and how to get the best out of it. In particular, such a person will know what to do when things go wrong, as they may from time to time.

One way to ensure that you have such a person is to choose a bright member of staff and give that member a machine to play with at home. If you have chosen the right person your apparent generosity will be amply repaid by the fund of knowledge and really infectious enthusiasm which can be generated. It is not generally appreciated by the unaddicted just how habit-forming playing around with computers may become. There are a number of people who program all night when they get carried away!

This is all to the good, when one considers the excessive expectations which can build up for the reliability of microcomputer programs.

It is a paradox that people using mainframe computers and paying hundreds of housands of pounds for them, are quite accustomed to paying tens of thousands of pounds for software and having the producers produce updates of those programs, correcting errors and making improvements, at frequent intervals. Thus they expect the programs to be far less than perfect and to be constantly revised. The first-time microcomputer user, however, is probably accustomed to all the electronic equipment he has been in contact with, from transistor radios to hi-fi systems, working perfectly and consequently expects computer programs to be just as reliable. In fact, computer programming is a very exacting task. It calls for a rather unusual set of abilities in the programmer. Imagination to forsee whatever can happen and plan for it. Discipline to be able to plan a complex project an bring it to a satisfactoy conclusion. Working with a machine which takes nothing for granted and must be told what to do. Where to find the information to work on and where to put the results. It is not surprising, in these circumstances that programs are not perfect, in fact what is surprising is how many amazingly effective programs are available for the machines at such low cost.

The usefulness of your in-house enthusiastic expert is that occasionally the machine will appear to go wrong, due to a slight anomoly in the way the program reacts to a situation unforseen by the programmer and a knowledgable operator will be able to remove the difficulty in a moment and be back in operation immediately, whereas the unitiated will puzzle for some time.

It is now possible to sign up for a "hot line " to be available, enabling you to telephone once per day with any query about your installation. This can be a real time-saver in times of difficulty. Totally inexperienced users could sign up for more than one subscription!

It is worth considering how vital your machine has or will become to the operation of your business and to invest in a service contract which will enable you to have your machine repaired in a very short time, perhaps with the temporary loan of other equipment if necessary. A more spectacular alternative is to invest in a spare system. This sounds very extravagent, but need not be so. You will tend to find that the abilities of the system are such that the suitable applications tend to multiply and you eventually run out of computer time for all the items you would like to put onto it. Also, there is the matter of staff training to be considered and the ability to use two machines at the same time for different jobs is not to be sneezed at. You may also find it useful to run trials of a proposed new program before committing yourself to its use.

To change the subject slightly; it will be readily appreciated that a loss of power during a computer run will cause you considerable aggravation and the need for a rerun of the work at present being carried out. (It will depend on how well organised your procedures are how much of a problem this will be for you and you should make sure that your dealer explains all about "back-up" procedures to prevent real inconvenience.)

Some parts of the country are subject to considerable fluctuations in mains voltages and in the continuity of electricity supply. If your area is one of these, you should very seriously consider installing a device for smoothing out the voltages and/or having a temporary battery supply which, if mains electricity fails, will enable the work being done to be closed down in an orderly fashion. These devices are not expensive compared with their time-saving potential.

One should consider the fact that the basic equipment is so cheap and the programs such bargains, that additional expenditure to make the life of the staff, who are the real users, a little smoother, is well worth while. This includes the provision of a satisfactory environment in which to use the machine: desks at the right height, proper typist's chairs, suitable lighting, well-designed stands to hold copy, appropriate storage facilities for diskettes and for binding and filing computer printout. These sound like comparatively unimportant matters, but they go a long way towards the happy acceptance of the new system and the contented working of the employees most closely involved.

It is important to make sure that no member of staff works too long at the machine in any one continuous period, or in one day. Short periods are the most effective, for both comfort and efficiency. It is certainly the case that there are varying degrees of tolerance to looking at the screen for long periods and short periods avoid eye-strain and the possibility of headaches.

It is very likely that the use of the system will cause a review of your business systems to be unavoidable and some change will prove necessary. Whilst the transition from other methods to the use of a computer should be as smooth as we can possibly make it, the design of programs will inevitably call for modification of how things are done at present. This is not necessarily a bad thing. Surely one of the worst reasons for carrying out a task in a particular fashion is "We've always done it that way". The review of systems which our computer now forces upon us may be a very good thing in giving an opportunity for a searching review into what is done, whether it should be done at all and if so, how it should be done. We may even find that the cost of the installation is recovered from the beneficial effects of this review, which is otherwise just the sort of thing to be put off "until we have time".

The likely outcome is that we are forced to rationalise our procedures and to put them into a logical order, with cross checks on accuracy as we go along. The fact that the computer is a rational logical robot, which does exactly what it is told, no more and no less, will have its effect. We will probably be forced to regularise ad hoc methods of dealing with exceptional situations.
It is always a problem to decide which applications to put onto your computer first. Perhaps you had a particular application in mind when buying the system, which is the best way to approach the selection of equipment. However a number of business men, convinced of the virtues of the microcomputer, buy a system with no firm idea as to exactly what it will be used for, but rather with a general feeling say about "Doing a bit of payroll and some accounting". We should appreciate that there are some applications which are highly suitable for the use of a computer and some which are not.

Fortunately the equipment itself is so cheap that it is comparatively easy for the investment to be justified by the use of a single program for a well-chosen application. It is possible to view all further applications as being cost-free (although some of my colleagues in the accounting profession may disagree on this method of cost allocation!). At least we may say, that at a time when average earnings exceed £6,000 a year, relatively small time-savings will pay for the system very rapidly.

For example, a number of payroll packages exist and you should be able to find one suitable for your application from amongst these. This raises the problem of how you set about ensuring that the program you buy for any application will prove to be suitable. The secret is not to buy the program as such, but rather to buy what it will do for your business. Approach your dealer with some sample data and do not ask for 'a payroll program' and seek to evaluate it for yourself. Instead tell the dealer how many employees you have, how many pay rates, how many bonus schemes, how many pension schemes, how many savings schemes, how many different methods of payment (cash, cheque, bank credit transfer, National Giro cheque or transfer). You should also specify whether you require the program to generate a list of coins and notes required for cash payments. The $64,000 question to the dealer is now 'Have you a program which do all this for me and which is capable of dealing with any forseeable expansion in the numbers we employ'. The dealer will either be able to oblige or not. If he offers to write such a program in a few days, my advice to you is to go elsewhwere, because this is far from being a tivial programming task!

Let us assume however that he offers such a program. Should we now buy it? Emphatically not; or at least not yet!. We must see it working. If the dealer is unwilling to demonstrate it for us, then, again, go elsewhere. The next stage is to persuade the dealer, not to demonstrate it , but rather to permit you, or better still, you and the member(s) of your staff who will be using the program, to try to operate the program. The danger of allowing the dealer to demonstrate is that there is always the possibility of the demonstration's accidentally glossing over flaws in the program which are likely to cause problems later!

You have brought the staff with you, not merely to encourage a feeling of consultation and involvement, but also for this next vital stage in the proceedings. You should imagine yourself to be testing the program for some Microcomputing Consumers Association, or acting like the Devil's Advocate when the question of whether a certain person should become a saint is being determined: You should look for all the possible ways of making the program malfunction. This involves trying to make every possible error in, say, half an hour, which you could reasonably expect to make in a year's normal operation. This is not suggested in any frivolous way, nor indeed in a mischievous fashion. The intention is entirely serious. Only in this way can you assess the likely outcome of using this program in your business.

It is extremely unlikely that the programmer will have managed to think of all the stupid things which you may be able to contrive to do when you are trying to be awkward!!: for instance taking a disk out if the machine tells you to ensure that you have inserted one, or deliberately putting it in the wrong drive; however you will be able to assess whether the program handles most likely errors and whether the ones that it cannot handle are such that you can live with them.
At this point you are able to buy the program, pausing only to make sure that changes in the Tax rules will be the subject of an updated version of it.

LETTERS

Dear Mr Middleton,

Many thanks for the disk copy of Adventure and your offer for financial help.

Our 'local' group covers a radius of thirty miles from Taunton in the south to Stow in the Wold in the north and we meet once a month.

The meetings are informal and generally consist of information swapping, solving machine code problems and generally anything we consider bringing up. Visitors are welcome but please ring first in case we have a space problem. All meetings are held at my home:-
23 Shepherd Haze, Wotton-Under-Edge
(phone 2498), Glocestershire

The date for the next few meetings are:- December 19, January 30th, February 27th, March 27th. All on a Friday and all starting at 8.00pm.

We have many uses for our PETs including data logging, teaching, electronics, small business uses.

Yours sincerely,
Mrs Janet Rich

Editors comment:-

The financial assistance to user groups consists of a £25.00 handout, with no strings attached, from the Commodore PET User Club and the cheque will be winging its way to the Region 'B' IPUG within the next few weeks. There is only £250.00 pounds available which is going to be distributed in £25.00 sums. So user groups its up to you. Details of your clubs activities and number of members should be supplied and don't forget there is the superb disk based Adventure which is free as well to User Groups.

A letter from Jim Butterfield

SUPERMON Bug report:-

Versions 2 and 4 of SUPERMON may give trouble, particularly with the tiny assembler, due to an error in the coding.

To fix the bug type:-

```
     LOAD"SUPERMON",8
     ?PEEK(1781)
```

If the PET returns a value of 26 then this confirms you have version 2. Now type:-

```
     POKE1781,6
     SAVE"@0:SUPERMON",8
```

This repair to SUPERMON is important; amazingly I've received no complaints and ran across the problem myself while showing somebody else how to use it.

Note on ADVENTURE: It's incompatible with some versions of the DOS wedge, if anybody has trouble running the program ensure that DOS is not loaded. I included a bootstrap program at the beginning of the ADVENTURE disk called ADVBOOT. This allows you to start up with LOAD"*",8 followed by RUN. I'm not sure what the nature of the problem is since ADVENTURE is 99% 'up-front' BASIC and does not use wedge techniques itself.

A recent discovery. If you somehow get an unclosed file (one with an asterisk in the directory), DO NOT SCRATCH IT. Get it out of the disk with a VERIFY (Collect). Scratching can potentially damage other files on the disk.

As long as I am on the subject of disks it is worth restating the two and a half major DON'TS on disk.

1. If you somehow try to write on a protected disk (write protect tab over the hole) do not do anything further until you power down the drives. Your protected disk will not be hurt but the system will very likly clobber some other disk. Turn the drives off and start again.

2. DON'T ever allow two disks with the same ID to be in the same work area. If you must do a backup (DUPLICATE command) get the other disk out of the work area - to another room, another building, another town. Serious trouble will occur to a disk's organisation if it is plugged into a drive that has been handling disks with the same ID. More backup disks have been ruined this way... and if you wreck your backups what do you do next?

2.5 Using @ for replacement of programs or files is still under a shadow. It seems to be OK on programs but shaky on sequential files. The cause is not known; trouble seems to strike at random and as far as I know is not reproducible. My personal suspicion is that it's associated with improperly closed files but that is only a guess. It would be best to steer clear of this (scratch or rename the old file) until the question is cleared up.

That all for now. Best regards,
Jim.

Dear Mr Middleton,

I was delighted to see in CPUCN that you are a Mechanical Engineer. I too am a Mechanical Engineer but if World War 2 had not intervened I would probably have ended up being an Electronics Engineer.

I bought an 8k 2ØØ1 series PET in the very early days and have been plagued by certain problems ever since. While I have learnt to deal with most of these, they do not go away and usually choose the most awkward moment to raise their ugly heads.

1) Since most of the problems reside in the ROMs the obvious solution would be to retrofit BASIC2 but I was quoted over 1ØØ pounds, is this correct?

2) I am under the impression that the PET clears all RAM when powering up, why then do the cassette buffers contain garbage?

3) Why, when reading data from the 2nd Cassette does it sometimes stop reading leaving the deck running continuously?

4) I have expanded my RAM with a 'Petite', so I have 4Øk extending into the Expansion RAM area. Could I use this to upgrade a 3ØØ8N? I seem to remember reading that the memory expansion socket is not the same as on the 2ØØ1-8.

5) I believe that if I am to get disks I shall need BASIC2 ROMs. Is this true?

Finally I would add that Commodore really cannot expect dealers to answer questions like these. I have been in business long enough to know that if you wnat to survive, you must put your effort where the main chance is, which for a dealer is selling up to date hardware. However, for someone like myself who is not given to playing games with a machine and therefor uses the equipment almost solely for business purposes these are the sort of problems which are a real pain in the neck and if COmmodore isolates itself from its customers and does not deal with them no-one else will, thus Commodores reputation will suffer, which is presumably why the User Club was started.

Yours sincerely,
T.P.Gordon Brown

Editor replys:

The only reason I went into engineering was that it was something I knew about. It seems strange but only five years ago the only computers where big expensive mainframes and before I went to Polytechnic the nearest I had come to a computer was owning a Sinclair Programmable! Once at Poly I spent a vast number of hours programming, luckily the language I chose was BASIC. By the time I found that I was addicted to computers I was too far into the engineering course to make it worth while changing over to computer science. When I later put my hands on a PET I lost my interest in the DEC mainframe instantly and when out and bought my own machine.

The 2ØØ1 series being the first generation machine was bound to have a few problems associated with it but as you point out these can usually be programmed around. Here are the answers to your questions.

1) The ROM retrofit for the 2ØØ1 to upgrade it to BASIC2 was to cost over £1ØØ but it has been set at £38 + VAT to be the same as the BASIC4 retrofit.

2) The PET as part of its power up routine tries to write 17Ø into every memory location starting at 1Ø24, when it does not read back 17Ø it assumes that it has reached the top of memory and sets pointers accordingly. The cassette buffers are below this address and are thus not affected by the power up routine.

3) There is a bug in the timing constants associated with data files. The answer was given in Volume 1 issue 1 and also in the 'Best of CPUCN' which costs £1Ø.ØØ and will be available shortly as it is at the printers now.

4) You will need to change the socket but the Petite will work with a 3ØØ8N.

5) The ROMs are not included in with the disk system but will be given free if you ask for them.

Dear Mr Middeton,

There are a few of points that arise from newsletter 2.8 which I have listed below.

1) I was pleased to read that I have won one of the 50 pound software prizes, as I already have most of the software in the Master Library, could I have a set of upgrade ROMs for my 3032 thus giving me access to BASIC4?

2) On page 8 Robert Oei points out a method of switching the printer to permanent lower case, thus allowing programs to be listed in lower case. He has highlighted a very serious bug that exists in the printer which appears only to be associated with ROM No 4, the latest ROM released for the printer.

This 04 ROM has an additional secondary address, number 7, which allows the user to switch the printer permanently into lower case, thereby avoiding the need for a cursor down at the start of every line. All that is needed is OPEN7,4,7:PRINT#7 and the printer will be switched. There appears to be no way to restore this mode to normal other than by switching the printer off and on again.

What Mr Oei has discovered is the fact that the flag that is set by the secondary address 7 is also accidentally set if, while a line of data is being printed under format control another formatted line is sent to the printer. The exact details are rather obscure and I am currently examining a disassembly of the ROM to try and find out where the problem arises. This bug can cause havoc if while trying to prepare formatted printouts the timing is such that one line is being sent to the printer at the same time as another is being printed thereby forcing all the output to be printed in lower case!

3) There is no doubt that the newsletter has come a long way since those early issues. However, I would like to suggest that items be delimited in some way. It is very difficult to identify where an item starts and finishes, making browsing through the newsletter almost impossible. Perhaps item heading could be in bold type while sub-headings remain in small type.

4) In CPUCN2.7 on page 2 there is a detail stating the undesirability of using the 'fast screen' POKE (ie. POKE59458,62) on the newer PETs. I have tried this on my new 3032 and found that the screen is reduced to only 20 scan lines (instead of 200) thus showing the first two and a half lines of characters. In addition, the interrupt appears to be speeded up significantly (about three times).

Looking at the circuit diagrams I can see there appears to be circuitry to perform such an operation. One wonders if this is just part of the address multiplexing or if it is something provided for future expansion? A comment in the next newsletter would be welcomed by several readers.

5) Several sources have given details of the Jim Butterfield 'uncrash' techinique of grounding the diagnostic pin and then performing RESET with a reset key. However, if a machine code program has crashed, very often the CHRGET routine at $0070 is corrupted, so in addition to typing a semicolon and getting a question mark in reply type:-

```
.M E0F9 E110
```

Move the cursor over the address and type 0070/0078/0080 for each of the three lines, pressing return each time of course. This will restore the CHARGET routine to normal. Type 'X' and return to BASIC then type 'X' again which should result in the message 'SYNTAX ERROR' being displayed. This last step ensures that the stack pointer will be restored correctly.

This technique will allow recovery from machine code crashes more frequently than the straightforward Butterfield 'un-crash'.

Yours sincerely,
Mike Todd

Editors reply:

Thanks for the letter Mike, a few comments back:-

1) Yes, you can have the ROMs when they are available, it will be a few weeks yet before I can send them to you as the ROMs are arriving in 4032 PETs rather than as retrofits.

3) Vol 3.1 fitted in with your comments I think, I am continuing the process with this issue, if anybody has suggestions for improvements they are always welcome. Would contributors sending in files on WordPro please use the following page layout: lm10:rm50:jul:lfl with the same paragraphing style as shown in the rest of the magazine.

4) Do you want to know a really good way of causing permanent damage to your PET? Try POKE59458,62. What happens is, you get two chips, a MOS and a low power Schokty fighting each other. The MOS wins and the Schokty dies leaving the PET with only two and a half lines of characters! You have been warned!

Unfortunatly there is nothing fantastic about the two data lines, Dave Briggs, Technical Support Manager says they are for address multiplexing.

BASIC PROGRAMMING

BASIC and DOS Conventions

David Middleton

As a convention Old and New ROM PET was quite satisfactory but with the arrival of BASIC4 there are now three versions of Microsoft BASIC, this means that 'Old' and 'New' ROM is no longer sufficient to describe which BASIC is being used. I am going to use the following conventions and I would advise authors to do the same.

BASIC1 the original language, standard in 8k small keyboard PET, formally called Old ROM.
BASIC2 the version of the language in the large keyboard PET, 3ØØØ series machines, formally called New ROM.
BASIC4 the version of the language used in the 8ØØØ series PET and also in the 4ØØØ series. Note that the difference between the 4ØØØ and the 8ØØØ series is in the screen editor and not the BASIC interpreter ie. the 4ØØØ series has the same screen editor as the 3ØØØ series. BASIC4 is available as a retrofit for BASIC2 machines.

Also it may now be necessary to describe which DOS you are using. I would suggest the following:-

DOS1 The version used in the 3Ø4Ø disk units as standard.
DOS2 Fitted to the 4Ø4Ø drives and supports relative records, available as a retrofit for 3Ø4Ø.
DOS2.5 Only available on the 8Ø5Ø disk system. Disks are totally incompatible with the 3Ø4Ø and 4Ø4Ø. Utilities are available to copy files from one disk system to another.

Remember that disks written on a DOS1 system can be read by DOS2 and vice-versa but no attempt should be made to write on a disk produced by the other DOS as corruption will occur in great proportions!

Hints on converting programs from BASIC 2.0 to BASIC 4.0 (40 column)

Paul Higginbottom

The best way I found to convert programs, was to divide all of the programs into four categories. These are as follows:-

1. Programs written entirely in BASIC, with no PEEK, POKE, USR, WAIT or SYS statements.

2. Programs written entirely in BASIC, with PEEK, POKE, USR, WAIT and/or SYS statements.

3. Programs written partly in BASIC and partly in machine code, with PEEK, POKE, USR, WAIT or SYS statements.

4. Programs written entirely in machine code.

Firstly I would like to discuss the utilities I use when converting programs.

I use BASIC AID for the BASIC conversion. This has FIND, CHANGE (something the TOOLKIT lacks), NUMBER (renumber), KILL (to exit), DELETE, and BREAK (drops you into the monitor). This is a BUTTERFIELD abbreviation of our own BASIC AID, MPØ96 (now on sale for 1Ø pounds! and has 25 commands - I think), but for BASIC 4.Ø. Also I use SUPERMON4.REL (by BUTTERFIELD/WOZNIAK/SEILER/QUITE A FEW OTHERS) which is an add-on to the monitor commands for BASIC4.Ø, allowing you to hunt for code or text, disassemble, assemble, list memory in ASCII as well as hex, step through programs with trace or step, etc. I use a disk unit for conversion, but I should think a tape user could do the same sort of thing (ONLY SLOWER). The memory maps mentioned below have been published in CPUCN Vol. 3, Issue 1 and will also be published in 'Best of CPUCN' - Soon to be published.

Now I will go through each category, one at a time.

1. This category shouldn't need any conversion.

2. Let's take the POKE statements first. Apart from those used to alter the screen RAM (which stay the same), usually the corresponding locations from machine to machine can be found by looking at Jim Butterfield's memory maps. The only other problems that seem to arise, are when a location has been poked with a certain value to make the PET function in a different way. A good example of this is the well known one that will disable the RUN/STOP key. If you can understand why it works, then conversion to BASIC 4.Ø is easy. All that is necessary, is to add three to the current contents of 144. On a 2.Ø Pet POKE144,49 will disable the stop key. This is three more than its normal contents (46). Therefore POKE144,PEEK(144)+3 would work on either machine. Just to save you the bother, it is in fact POKE144,88 (to disable), and

POKE144,85 (to enable), on BASIC 4.0 machines.

If the program is entirely BASIC, then the USR and SYS commands will not be used (unless routines from the ROMs are being used). If ROM routines are being used, again memory maps are necessary.

The WAIT command is generally only used for keyboard activity: WAIT152,1 (wait for shift key), and WAIT158,1 (wait until bit 0 of the number of keypresses in the buffer is a 1; i.e wait until an odd number of keypresses > 0). The two just mentioned would be the same on 2.0 and 4.0.

The USR command would only be used if machine code was also used, but that is not covered in this category.

3. All hints made in category 2 should be observed for this category as well. The USR command uses bytes 1 and 2 as an indirect address to a machine code routine. The parameter in the USR command is 'floated' and put into the first accumulator. The address POKE'd into the bytes 1 and 2 will obviously not need to be changed but the actual machine code routines, will more than likely need to be changed. The routines most commonly used by USR routines are FLPINT (floating point to integer conversion for accumulator #1) and of course INTFLP (the other way round!). The corresponding locations can again be found in the Butterfield memory maps. Use FIND/POKE1/ to find the USR command set-up statements and work out the hex address. (See Dec to Hex conversion in this issue). Use SUPERMON to disassemble the USR code and make any changes on the screen (JMP's into ROM usually). You should also know where your program starts in memory. To find this out from a disk unit on a BASIC 4.0 machine, the following program will do:-

```
10 INPUT"FILENAME";F$:INPUT"DRIVE";DR
20 DOPEN#1,(F$),D(DR):IF DS THEN
   PRINTDS$:GOTO60
30 GET#1,A$,B$:N$=CHR$(0)
40 AD=ASC(A$+N$)+ASC(B$+N$)*256
50 PRINT"PROGRAM STARTS AT"AD
60 DCLOSE#1
```

You may want to add a little hex converter into the program.

To resave programs that do not start at $0401/1025, you would need to drop into the monitor (SYS4 for example). Then you would need to see where your program ends by typing in .M 002A 002A <RETURN>. The contents of 002A,002B are the end of your program (LOW, HIGH). Let us say for example that:-

```
.:  002A 40 1B 40 1B 40 1B 00 00
```

appears. To save your program onto drive 0 on disk, you would need to type:-

```
.S "0:FILENAME",08,033A,1B41
```

033A is the start address.
1B41 is the end address (1 More than necessary, because the monitor does not save the last byte!)

4. Programs written entirely in machine code usually fall into three categories.

(i) Those that use ROM entry points, and system variables all over the place.
(ii) Those that only use system variables (keyboard usually).
(iii) Those that manage everything by themselves.

As before, I will handle each case separately.

(i) Tiresome, because usually the whole program will have to be disassembled onto paper, and the listing gone through with a pen, whilst clutching memory maps!

(ii) Shouldn't be too much trouble, since most system variables are the same.
NOTE:- $97 (151) = Keyboard Matrix coordinate on graphics keyboards,
= Unshifted ASCII on business keyboards.

(iii) Will almost certainly work. Only keyboard type may cause problems.

Adding commands to BASIC

David Simons

Trying to add commands to BASIC is difficult and good knowledge of 6502 assembler language is required.

To add a command you need to know about the CHARGET (character get) routine which starts at $70, the routine is used to scan BASIC lines and the first few bytes of it reads:-

```
;increment low source
 0070 E6 77  INC $77
;new page ??
 0072 D0 02  BNE $0076
;yes increment page counter
 0074 E6 78  INC $78
 0076 ;more code......
```

It then continues to determine if the current character is a space or colon but all you need to know is the above. Quite simply you change it to a jump statement so if your routine starts at $7700 then all you have to do is type:-

```
 0070 4C 00 77  JMP $7700 ;Goto $7700
```

(4C is JMP in 6502 machine code). To set the start address to which the PET is to jump you can use the routine below:-

Any start address may be chosen for the enable routine below but remember to work out the address in decimal so you know the number to use with the SYS command. Remember if you are going to use a BASIC loader program you will not want to load separate areas of memory so the best approach is to put the enable routine (as below) before the routine for your new commands. The code shown below is relocatable anywhere, $033A is the start of the 2nd cassette buffer.

```
 033A  A9 00  LDA #$00
;Load the accumulator with the
;low byte of the routine.
;Then store it.
 033C  85 71  STA $71
;Load the accumulator with JMP.
 033E  A9 4C  LDA #$4C
 0340  85 70  STA $70
;Load the accumulator with
;high byte of the routine.
 0342  A9 77  LDA #$77
 0344  85 72  STA $72
;Return to BASIC.
 0346  60     RTS
```

To use it type SYS 826, NEVER try to change the CHARGET routine using POKE - you will have to reset the PET if you do!! Don't try this routine until you have written your command(s) and saved a copy of them.

Once you have the PET going to your routine you have to use a short piece of machine code to decipher whether the PET is meant to be executing the command or storing it in memory. This is achieved like this:-

```
;store x register
 7F00  8E 3A 03  STX $033A
;transfer stack pointer to X register
 7F03  BA        TSX
;is the PET in edit mode?
 7F04  BD 01 01  LDA $0101,X
 7F07  C9 F9     CMP #$F9
 7F09  D0 10     BNE $7F1B
 7F0B  BD 02 01  LDA $0102,X
 7F0E  C5 C6     CMP $C6
 7F10  F0 09     BEQ $7F1B
;yes-increment the source by 1
 7F12  E6 77     INC $77
 7F14  D0 02     BNE $7F18
 7F16  E6 78     INC $78
;go and see if it is a new command
 7F18  4C 7F 27  JMP $7F27
;no-Get x register back
 7F1B  AE 3A 03  LDX $033A
;increment source
 7F1E  E6 77     INC $77
 7F20  D0 02     BNE $7724
 7F22  E6 78     INC $78
;get next byte.
 7F24  4C 76 00  JMP $0076
;get ready to see if it is a new command
 7F27  A0 00     LDY #$00
;Get current character
 7F29  B1 77     LDA ($77),Y
;check to see if it is a new command.
 7F2B  C9 40     CMP #$40
```

We will stop the listing there for a while. What you have to do now is to decide what character you want to use as a start for your commands, there is not a big choice, these are the ones you can use @ , ! , ', &, [,] ,and $. Once you have decided on a character at address $772B change the 40 after C9 to hex of the ASCII of the character you have chosen.

The listing continues :-

```
 7F2D  F0 03    BEQ $7F30;
;Not a new command therefore
;jump to reget.
 7F2F  4C 76 00 JMP $0076
;increment source low byte.
 7F32  E6 77    INC $7
 7F34  D0 02    BNE $7F38
;increment source hi byte.
 7F36  E6 78    INC $78
;get new byte
 7F38  B1 77    LDA ($77),Y
;see if it is the correct letter
;for a new command
 7F3A  C9 50    CMP #$50
 7F3C  F0 26    BEQ $7F64
;Not a new command goto reget
 7F3E  4C 76 00 JMP $0076
```

You have to decide on a suitable character for your new command. Eg. if it inverts the screen a good letter would be 'I'. Place the hex of the ASCII of the required letter after the C9 at the address $7F3A. The jump at $7F3E can be to $0076 but if you have another command change the $0076 to the address where that command starts. Then all you have to do is compare it with another suitable letter and so on.

Evaluating parameters.

Many commands need parameters and there are routines in the interpreter which can be used:

$D675 - Returns the number in the x register, but if the number is greater than 255 it will print an error, so for numbers greater than 255 you must use the following routines : $CC8B followed by $D6D2
You will then find the low byte of the number in the y register and the high byte of the number in the accumulator.

$D6CC - Returns the number in the x register, again if the number is above 255 use $CC8B and $D6D2. This routine will also return an error if no comma is put before the parameter.

$F4FD - Evaluates a string. The string's position in memory is held in the x register (low byte) and the accumulator (hi byte). The length is stored in $D1.

$CDF8 - Checks for a comma, if there isn't one it returns a "?SYNTAX ERROR". Thus you can transfer as many parameters as you like to your routine.

Printing Results.

$CA1C - Will print a string, the accumulator must hold the low byte of the address where the string starts and the Y register must hold the high byte. You must end the string with $00 otherwise the PET will jam.

$FFD2 - Will print the character held in the accumulator.

Getting Input.

$FFCF - Will return one character (like GET) this character can be found in the accumulator. If no keys have been pressed it will return $00.

$FFE4 - Will do exactly the same as the above routine but will not display a cursor.

Zero Page.

For indirect addressing zero page is used, the following locations will not affect the PET; $00, $01, $02, $0F, $10, $21, $22. $21 and $22 will not stay intact as some PET routines use them and remember do not use USR if you are using locations $00, $01 or $02.

To Halt BASIC Execution

$C389 - Will stop BASIC and display "READY."

In CPUCN Vol 2 Issue 8 a 'RESTORE DATA LINE' program was described. By using the program you find that the hi and low bytes of the line number you want have to be poked into memory but if we make this a BASIC command it can be used simply (ie you don't POKE in the line number but you just follow the command with the required line number). The listing below when added to the previous listing ($7F00 to $7F3E) will accomplish this task. For people without an assembler or who do not use machine code but want the command they should use the BASIC loader program at the end. See the BASIC loader program for instructions.

Data Restore Routine

```
7F41 A6 3C    LDX $3C
7F43 86 11    STX $11
7F45 A6 3D    LDX $3D
7F47 86 12    STX $12
7F49 20 2C C5 JSR $C52C
7F4C 90 0B    BCC $7F59
7F4E A6 5C    LDX $5C
7F50 8E 94 03 STX $0394
7F53 A6 5D    LDX $5D
7F55 8E 95 03 STX $0395
7F58 60       RTS
7F59 A2 00    LDX #$00
7F5B 8E 94 03 STX $0394
7F5E A2 00    LDX #$00
7F60 2A       ROL
7F61 85 03    STA $03
7F63 60       RTS ;end of data restore

   ;start of command

7F64 20 9D 7F JSR $7F9D
7F67 20 D2 D6 JSR $D6D2
7F6A 85 3D    STA $3D
7F6C 86 3C    STY $3C
7F6E 20 41 7F JSR $7F41
7F71 AD 94 03 LDA $0394 ;error ??
7F74 6D 95 03 ADC $0395
7F77 C9 00    CMP #$00
7F79 D0 03    BNE $7F7E
7F7B 4C 89 C3 JMP $C389 ;Yes-goto READY.
7F7E AD 94 03 LDA 0394  ;No-continue
7F81 85 3E    STA $3E
7F83 AD 95 03 LDA $0395
7F86 85 3F    STA $3F
7F88 A5 3E    LDA $3E
7F8A C9 00    CMP #$00
7F8C F0 05    BEQ 7F93
7F8E C6 3E    DEC #$3E
7F90 4C 76 00 JMP $0076
7F93 C6 3E    DEC $3E
7F95 C6 3F    DEC $3F
7F97 4C 76 00 JMP $0076
7F9A C6 3E    DEC $3E
7F9C 60       RTS
7F9D E6 77    INC $77   ;get ready to
7F9F D0 02    BNE $7FA3 ;take parameter
7FA1 E6 78    INC $78
7FA3 20 8B CC JSR $CC8B ;get parameter
7FA6 60       RTS
7FA7 A9 4C    LDA #$4C ;enable command
7FA9 85 70    STA $70
7FAB A9 00    LDA #$00
```

```
7FAD 85 71    STA $71
7FAF A9 77    LDA #$7F
7FB1 85 72    STA $72
7FB3 A9 7E    LDA #$7E ;protect memory
7FB5 85 35    STA $35
7FB7 85 31    STA $31
7F9B A9 FF    LDA #$FF
7FBB 85 3Ø    STA $3Ø
7FBD 85 34    STA $34
7FBF 4C 89 C3 JMP $C389; Jump to READY.
```

Finaly perhaps a few ideas....

The 2nd letter (ie the one that comes after the @,$,etc) can be changed so that the number is a BASIC token. So you can have @RESTORE for the above program.

The CHARGET routine could be made to jump to a trace routine , step etc.

Never enable new command(s) without having made a copy on a tape or disk. This applies to all machine code programs as machine code has the habit of jamming the PET when not properly entered. If you don't use the 2nd cassette buffer you will have to protect your routine from BASIC. The best way of doing this is to put the machine code at the top of memory and then POKE 52,low of beginning of command(s) :POKE 53, high of beginning of command(s) and then repeat for locations 49(same as 53) and 48(same as 52).

If you have a command that will take a lot of machine code statements it is advisable to write the code needed, check it works, and then interface it to BASIC.

PROGRAM NAME: DATA RESTORE

```
1ØØ PRINT"              COMMAND TO RESTORE
    DATA":PRINT
11Ø PRINT"FOR BASIC2 32K PETS":PRINT
12Ø PRINT"RESTORE ROUTINE BY  : PAUL BARNE
    S":PRINT
13Ø PRINT"COMMAND ADDED BY    : DAVID SIMO
    NS":PRINT
14Ø PRINT"TO ENABLE           : SYS 32679"
    :PRINT
15Ø PRINT"TO USE              : @P X":
    PRINT:PRINT:PRINT
151 PRINT"WHERE X = THE LINE TO WHERE
152 PRINT"THE DATA IS TO BE RESTORED"
16Ø POKE49,126:POKE53,126:PRINT:PRINT"LOA
    DING"
17Ø FORT=32512TO327Ø5:READA:POKET,A:NEXT:
    PRINT"FINISHED":
18Ø DATA142,58,3,186,189,1,1,2Ø1,249,2Ø8,
    16,189,2,1,197,198,24Ø,9,23Ø
19Ø DATA119,2Ø8,2,23Ø,12Ø,76,39,127,174,5
    8,3,23Ø,119,2Ø8,2,23Ø,12Ø,76
2ØØ DATA118,Ø,16Ø,Ø,177,119,2Ø1,64,24Ø,3,
    76,118,Ø,23Ø,119,2Ø8,2,23Ø,12Ø
21Ø DATA177,119,2Ø1,8Ø,24Ø,38,76,118,Ø,16
    6,6Ø,134,17,166,61,134,18,32
22Ø DATA44,197,144,11,166,92,142,148,3,16
    6,93,142,149,3,96,162,Ø,142
23Ø DATA148,3,162,Ø,42,133,3,96,32,157,12
    7,32,21Ø,214,133,61,132,6Ø,32
24Ø DATA65,127,173,148,3,1Ø9,149,3,2Ø1,Ø,
    2Ø8,3,76,137,195,173,148,3,133
25Ø DATA62,173,149,3,133,63,165,62,2Ø1,Ø,
    24Ø,5,198,62,76,118,Ø,198,62
26Ø DATA198,63,76,118,Ø,198,62,96,23Ø,119
    ,2Ø8,2,23Ø,12Ø,32,139,2Ø4,96
27Ø DATA169,76,133,112,169,Ø,133,113,169,
    127,133,114,169,126,133,53,133
28Ø DATA49,169,255,133,48,133,52,76,137,1
```

Formatting Listings

Dave Middleton

The idea for this program came from Mark Clarke while he was working for us at Commodore. One of the utilites which Mark wrote was for a program which would list another from disk without the program actually being in memory. The technique for getting a program to list from disk is fairly simple but I will give an indication of how programs are stored so that you can if you wish perform your own functions.

```
10 PRINT"HI"
20 END
```

The above program will be stored on disk in the following format:-

```
|ØØ1|ØØ4|Ø11|ØØ4|Ø1Ø|ØØØ|153|Ø34|Ø72|Ø73|
  START  |LINK AD|LINE NO| ?  | "  | H  | I  |

|Ø34|ØØØ|Ø17|ØØ4|Ø2Ø|ØØØ|128|ØØØ|ØØØ|ØØØ|
| " |NUL|LINK AD|LINE NO|END|NUL|NUL|NUL|

READY.
```

1. START: The first two bytes of the program stored on disk give the location of where it is to be put into memory (BASIC normally starts at 1Ø25)

2. LINK AD: Points to the next link.

3. LINE NO: BASIC line number.

4. BASIC statement: Keywords have a value between 128 and 2Ø7

5. NUL: End of BASIC statement.

6. Repeat from No.2 until the link address is Ø. When it is Ø this is the end of the program. Easy!

There is a two complications which are fairly simple to overcome. The BASIC keywords are saved as 1 byte tokens with a value in the range 128-2Ø7, a lookup table is used when converting these into a readable format for LISTing, for simplicity I use an array containing the keyword rather than using BASIC's lookup table. As everybody who used the screen editor knows, BASIC has a quotes mode which says that when ever quotes mode is on then every character received will be treated literally. Thus characters such as clear screen which has a code of 147 will be performed as clear screen and not translated into the keyword LOAD. Whenever a quote is encounted in a BASIC line a quote flag is set and then reset when another is found. The quotes mode is reset when a new line is started.

With the above information it is a fairly simple task to write a program which will read data from disk and print it to the screen of the PET. I wanted to perform other tasks than this, notably getting the column width to be the same as the column width used for CPUCN. This gives other problems. If you key in the program and run it you will find that quotes appear at odd intervals on the left hand side of the screen or printer. The reason for this is that the quotes mode is switched off when ever a carriage return is executed. It is necessary to switch quotes mode back on again so that reverse field characters will be printed rather than executed. Remember that the Commodore 3Ø22 printer will drop into lower case if a cursor down is recieved outside quotes.

```
 PROGRAM NAME: LIST

10 LW=48:X=0:NQ=1:OPEN15,8,15:OPEN3,3:E=2
   56:QU$=CHR$(34):  GOTO90
50 INPUT#15,ER,B$,TR,SE IFER=0THENRETURN
55 PRINTER,B$;TR;SE:GOTO100
90 GOSUB11500
100 PRINT"■■■OUTPUT FINISHED■ PRESS A KEY
     TO CONTINUE":POKE158,0
110 GETA$:IFA$=""GOTO110
120 RUN
9000 NO=75:DIMOP$(NO):FORI=0TONO:READOP$(
     I):NEXT::RETURN
9100 DATA END,FOR,NEXT,DATA,INPUT#,INPUT,
     DIM,READ,LET,GOTO,RUN,IF,RESTORE
9110 DATA GOSUB,RETURN,REM,STOP,ON,WAIT,L
     OAD,SAVE,VERIFY,DEF,POKE,PRINT#
9120 DATA PRINT,CONT,LIST,CLR,CMD,SYS,OPE
     N,CLOSE,GET,NEW,TAB(,TO,FN,SPC(,THEN
9130 DATA NOT,STEP,+,-,*,/,↑,AND,OR,>,=,<
     ,SGN,INT,ABS,USR,FRE,POS,SQR,RND,LOG
9140 DATA EXP,COS,SIN,TAN,ATN,PEEK,LEN,ST
     R$,VAL,ASC,CHR$,LEFT$,RIGHT$,MID$,GO
11500 PRINT"■LIST":B=30:PRINT"■■"
11502 CMD3,;: INPUT"FILENAME ? ";F$:F$=
      LEFT$(F$,15):PRINT
11504 CMD3,;: INPUT"DRIVE NUMBER ? ";A$:
      IFA$<"0"ORA$>"1"GOTO11504
11520 PRINT: OPEN2,8,0,A$+":"+F$+",PRG,RE
      AD":GOSUB50
11522 CH=3:CMD3,;:INPUT"■SCREEN OR PRINTER
       ? S■■";A$:PRINT: IFA$="P"THENCH=4
11524 OPEN4,CH
11525 GOSUB9000
11530 IFCH=3GOTO11538
11535 INPUT"PROGRAM NAME ? ";A$:PRINT#4,"
      PROGRAM NAME: ";A$:PRINT#4
11536 PRINT#4:PRINT#4:PRINT#4
11538 C=2:GOSUB12000
11540 LM=LN:Q=0:GOSUB12000:IFLN=0THEN
      PRINT:CLOSE2:GOTO100
11541 IFLEN(R$)>LEN(STR$(LM))+7THEN
      PRINT#4,R$
11545 GOSUB12000:R$="         "+STR$(LN)+"  "
11550 GET#2,A$:IFST<>0THENPRINTCR$;EF$:
      CLOSE2:GOTO100
11570 IFA$=""THENQ=0:GOTO11540
11573 IFA$=QU$THENQ=NOTQ
11580 IFASC(A$)=>128GOTO11590
11582 R$=R$+A$:IFLEN(R$)=>LWTHENPRINT#4,R
      $:GOSUB15000
11584 GOTO11550
11590 IFQORASC(A$)>203GOTO11582
11591 A$=OP$(ASC(A$)-128)
11592 IFLEN(R$+A$)=>LWTHENPRINT#4,R$:
      GOSUB15000:R$=R$+A$:GOTO11550
11593 R$=R$+A$:GOTO11550
12000 GET#C,A$:LN=0:IFA$<>""THENLN=ASC(A$
      )
12010 GET#C,A$:IFA$<>""THENLN=LN+E*ASC(A$
      )
12015 GETA$:IFA$="C"GOTO100
12020 RETURN
15000 R$=LEFT$("                          ",
      LEN(STR$(LN))+7)
15005 IFNQ=0GOTO15020
15010 IFQTHENR$=CHR$(34)+LEFT$(R$,LEN(R$)
      -1)
```

A Story

The following program should provide a bit of fun, especially for the unsuspecting user. It is one of the programs from a set of four disks which Nick Green has in his possession for the Education Software Workshops, the disks will be available to bona-fide organisers only.

The program is obviously an attempt to get kids to learn grammatical terms and as such is quite sucessful but wrong answers can give even more hilarious results than the correct ones!

```
1 poke 59468,14
2 print"This is a MAD LIB, you fill in th
  e "
4 print"missing parts of speech. Good luc
  k!"
5 rem written by pat tubbs 1978
6 print
1Ø rem mad lib - a story
2Ø print " Type an adjective":input a$
3Ø print " Type the name of a girl":
   input b$
4Ø print " Type an adjective":input c$
5Ø print" Type the name of a man":input d
   $
52 print"Type an adverb":input e$
54 print"Type a noun":input f$
6Ø print"Type a noun": input g$
7Ø print"Type an adverb":input h$
8Ø print" Type an exclamation":input i$
9Ø print" Type an adjective":input j$
1ØØ print" Type an adverb":input k$
11Ø print"Type an adjective":input l$
12Ø print"Type an adjective":input m$
13Ø print"Type a noun": input n$
135 print""
14Ø print"       A STORY          "
15Ø print" "
16Ø print"  There was a sudden "a$" knock
    "
17Ø print"on the door.  "b$" turned to "
18Ø print"her "c$" lover. 'Quick, "d$"',
19Ø print"she cried "e$", 'Hide in"
2ØØ print"the "f$" closet.'"
21Ø print"It was her boy friend. 'I forgo
    t"
22Ø print"my " g$"', he said "h$"."
23Ø print"'"i$"', he added. 'Where did th
    is "j$
24Ø print"cigar come from?'            "
25Ø print"'What cigar',she asked "k$"."
26Ø print"'Don't give me any of that "l$
27Ø print"stuff', he shouted. 'Next time"
28Ø print"I catch you smoking my cigars,"
29Ø print"I'm going to knock the "m$" "n$
```

Hex/Decimal Conversion

In Volume 2 Issue 8 I published a Hex to decimal conversion program written by R Brand. I suggested that this could be shortened to six lines. I had considerable correspondence on this and some of the results where very good but nobody could match Paul Higginbottom who producea the following code with only four lines ot BASIC.

```
100 INPUTA$:W=48:K=16:Q=LEN(A$):DEFFNT(
    T)=T+7*(T>9):IFASC(A$)<>36THEN120
110 FORI=0TOQ-2:T=ASC(MID$(A$,Q-I))-W:J
    =FNT(T):V=V+J*K↑I:NEXT:PRINTV:RUN
120 N=VAL(A$):PRINT"$";:D=-LOG(N)/LOG(K
    ):D%=D-(D<>INT(D)):FORI=D%TO0
130 P=K↑(-I):Q%=N/P:PRINTCHR$(Q%+W-7*(Q
    %>9));:N=N-Q%*P:NEXT:PRINT:RUN
```

A program to convert machine code to BASIC DATA Statements

Dave Middleton

I wrote the following program to enable me to convert machine code into DATA statements so that BASIC can read the decimal number and then POKE it into the correct place in memory. The method used is one of the few examples where a self writing program is actually useful.

The program works out which version of PET BASIC it is using so that it can access the correct area of memory set aside for the keystroke buffer.

```
10 INPUT"INPUT OLD OR NEW TYPE ROM O/N█
   ██D████";AN$:IFAN$="O"THENPE=1:GOTO40

20 IFAN$="N"THENPE=2:GOTO40
30 PRINT"INPUT O OR N DEPENDING ON TYPE
   ":GOTO10
40 INPUT"START ADDRESS IN DECIMAL";CO:
   INPUT"FINISH ADDRESS IN DECIMAL";EN
50 INPUT"STARTING LINE NUMBER ";LN
60 GOSUB150
70 CO=CO+1:V=PEEK(CO)
80 S$=STR$(V):A$=A$+RIGHT$(S$,LEN(S$)-1
   )
90 IFLEN(A$)<75ANDCO<ENTHENA$=A$+",":
   GOTO70
100 PRINT"████";A$:PRINT"CO=";CO;":EN="
    ;EN;":PE=";PE;":LN=";LN;":GOTO4570█
    "
110 IFPE=1THENPOKE525,2:POKE527,13:
    POKE528,13:END:REM OLD ROM
120 IFPE=2THENPOKE158,2:POKE623,13:
    POKE624,13:END:REM NEW ROM
130 IFCO<ENTHENLN=LN+10:GOSUB150:GOTO70

140 END
150 A$=STR$(LN)+"DATA":RETURN
```

MACHINE CODE

DIMP Revisited

Danny Doyle

Among the ideas put forward for further experimentation, it was stated that BASIC images could be dynamically built, or 'canned', in any memory area for further processing by DIMP. This, unfortunatly is not the case. In fact, any image to be processed by DIMP must reside in the BASIC input buffer, the start of which is at $0200. The reason being that the firmware routine that DIMP calls to convert keyword tokens expects the image to be in no other place than BASIC's input buffer. Alas such is the way of software. So if, for example you are planning to give some canned BASIC images in your assembler program, you will need a small routine to block transfer them to BASIC's input buffer prior to processing them with DIMP.

One other point needs mentioning. The original version of DIMP, more by accident than design, automatically handled multi-statement images such as:-

```
A$="BLA": B$="BLA": ?A$,B$
```

However, if you intend to replace the call to BASIC's input routine, line 5, by a call to your own image building code, you will lose the automatic multi-statement handling. No sweat, help is at hand. Listing 1 shows an updated version of DIMP which will take care of any multi-statement, regardless of the method used to place the image in BASIC's input buffer.

To finish, if you do substitute line 5 by a call to your own routine, remember that on return the X and Y registers must be pointing to the address minus one of BASIC's input buffer; ie. X reg. should contain $FF and the Y reg. $01.

```
LINE# LOC    CODE         LINE


0001  0000                                         ;DIMP VERSION2 BY DANNY DOYLE
0002  0000                                         ;
0003  0000               IMAGEL=$77
0004  0000               IMAGEH=$78
0005  0000               INPUT  =$C46F
0006  0000               CHRGET=$0070
0007  0000               TOKEN=$C495
0008  0000               SYNERR=$CE03
0009  0000               EXECUT =$C700
0010  0000               ;
0011  0000                      *=$033A
0012  033A               ;
0013  033A  A5 77               LDA IMAGEL      ;SAVE BASIC IMAGE PTR.
0014  033C  48                  PHA             ;
0015  033D  A5 78               LDA IMAGEH      ;
0016  033F  48                  PHA             ;
0017  0340  20 6F C4            JSR INPUT       ;BASIC INPUT ROUTINE
0018  0343  86 77               STX IMAGEL      ;X AND Y POINT TO THE IMAGE
0019  0345  84 78               STY IMAGEH      ;
0020  0347  20 70 00            JSR CHRGET      ;SCAN IMAGE
0021  034A  20 95 C4            JSR TOKEN       ;TOKENISE KEYWORDS
0022  034D  20 70 00     RESTOR JSR CHRGET      ;RESTORE IMAGE PTR.
0023  0350  20 00 C7            JSR EXECUT      ;EXECUTE THE STATEMENT
0024  0353  A0 00               LDY #$00        ;INITIALISE PTR
0025  0355  B1 77               LDA (IMAGEL),Y  ;NEXT IMAGE CHARACTER
0026  0357  D0 07               BNE COLON       ;BRANCH IF ":"
0027  0359  68                  PLA             ;ELSE RESTORE
0028  035A  85 78               STA IMAGEH      ;BASIC IMAGE PTR.
0029  035C  68                  PLA             ;
0030  035D  85 77               STA IMAGEL      ;
0031  035F  60                  RTS             ;RETURN TO USER
0032  0360  C9 3A        COLON  CMP #':'         ;REALLY A COLON?
0033  0362  F0 E9               BEQ RESTOR      ;YES, SCAN NEXT IMG PART
0034  0364  4C 03 CE            JMP SYNERR      ;NO, 'SYNTAX ERROR'
```

```
 10 REM AN EXAMPLE OF DIMP IN USE
 20 PRINT"YES":SYS826:IF A$<>"END"GOTO20
 30 PRINT"THANK YOU FOR USING DIMP"
READY.
```

A double density line plot routine

Dave Middleton

The PET normally can only plot to a resolution of 4Ø across by 25 down. This is obviously of little practical value and to compensate for this there have been quite a few programs for plotting on the screen using double density graphics. This brings the resolution to a more respectable 8Ø by 5Ø points. The routine I have used in the line plot has been taken from Nick Hampshires "The PET Revealed". I had a requirement for lines to be plotted using DDG so I wrote a BASIC program to draw lines from any two coordinate pairs. The only problem with using BASIC for something as repetitive as this is that it is terribly slow. Machine code is the answer and the program shown below is the result. The bit that does all the work starts at line ØØ76 and finishes at line 1Ø3, it is well commented and I leave it to you to work out what is going on, I would suggest working through a few example coordinates with pencil and paper as the easiest method for enlightenment.

I have also included a couple of BASIC programs which show routine in use. The program will works on BASIC2 and 4Ø column BASIC4 machines. To get it to run on the 8Ø column PET you will need to multiply the Y coordinate by 2 again at line 176 and alter the check in line 149. To get it to run on BASIC1 machines it will be necessary to change the zero page locations. I specifically located the routine at 74ØØ decimal so that BASIC1 users could load the code without having to recalculate the jump instructions. BASIC1 users will also have change locations $B7 through to $C2 so that the code does not trample on BASIC. I would suggest using the BASIC INPUT buffer ($ØA to $59).

```
5 XS=186:XF=188:YS=187:YF=189:AD=183
6 POKEAD,Ø
1ØØ  FORA=1TO12:READX1,Y1,X2,Y2:POKEXS,X1:
     POKEYS,Y1:POKEXF,X2:POKEYF,Y2
11Ø  SYS74ØØ:NEXT:RESTORE:IFPEEK(AD)=1
     THENPOKEAD,Ø:GOTO1ØØ
115  IFPEEK(AD)=ØTHENPOKEAD,1:GOTO1ØØ
2ØØ  DATA 1Ø,4Ø,1Ø,2Ø
21Ø  DATA 1Ø,4Ø,4Ø,4Ø
22Ø  DATA 4Ø,4Ø,4Ø,2Ø
23Ø  DATA 4Ø,2Ø,1Ø,2Ø
24Ø  DATA 1Ø,2Ø,3Ø,1Ø
25Ø  DATA 4Ø,2Ø,6Ø,1Ø
26Ø  DATA 3Ø,1Ø,6Ø,1Ø
27Ø  DATA 3Ø,1Ø,3Ø,3Ø
28Ø  DATA 3Ø,3Ø,6Ø,3Ø
29Ø  DATA 6Ø,3Ø,6Ø,1Ø
3ØØ  DATA 6Ø,3Ø,4Ø,4Ø
```

```
5 X1=186:X2=188:Y1=187:Y2=189:AD=183
6 XM=79:YM=49:XL=Ø:YL=Ø
7 FORC=2ØTOØSTEP-4
1Ø  POKEX2,4Ø:POKEY2,25
2Ø  POKEAD,Ø:FORA=ØTO2*STEP/(59-C)
3Ø  POKEX1,INT(SIN(A)*(C+14)+4Ø):POKEY1,
    INT(COS(A)*(C+5)+25):SYS74ØØ:NEXT
5Ø  POKEAD,1:FORA=ØTO2*STEP/(59-C)
6Ø  POKEX1,INT(SIN(A)*(C+12)+4Ø):POKEY1,
    INT(COS(A)*(C+3)+25):SYS74ØØ:NEXT
```

```
LINE# LOC   CODE        LINE

ØØØ1  ØØØØ              ;********************************
ØØØ2  ØØØØ              ;*PROGRAM TO PLOT LINES USING   *
ØØØ3  ØØØØ              ;*DOUBLE DENSITY GRAPHICS.      *
ØØØ4  ØØØØ              ;*                              *
ØØØ5  ØØØØ              ;*BY DAVE MIDDLETON             *
ØØØ6  ØØØØ              ;*                              *
ØØØ7  ØØØØ              ;*PUT Ø IN $B7 (183) TO ADD     *
ØØØ8  ØØØØ              ;*PUT 1 IN $B7 (183) TO DELETE  *
ØØØ9  ØØØØ              ;*XSTART  =$BA (186) X1         *
ØØ1Ø  ØØØØ              ;*YSTART  =$BB (187) Y1         *
ØØ11  ØØØØ              ;*XFINI   =$BC (188) X2         *
ØØ12  ØØØØ              ;*YFINI   =$BD (189) Y2         *
ØØ13  ØØØØ              ;********************************
ØØ14  ØØØØ                                     ;
ØØ15  ØØØØ                                     ;
ØØ16  ØØØØ                                     ;
ØØ17  ØØØØ                                     ;VARIABLES USED FOR DDP
ØØ18  ØØØØ                                     ;
```

```
ØØ19  ØØØØ                 XCOORD =$ØØ
ØØ2Ø  ØØØØ                 YCOORD =$Ø1
ØØ21  ØØØØ                 AORD   =$B7         ;ADD OR DELETE POINT
ØØ22  ØØØØ                 BINOFF =$B8
ØØ23  ØØØØ                 ERROR  =$B9
ØØ24  ØØØØ                                       ;
ØØ25  ØØØØ                                       ;VARIABLES USED FOR LINE PLOT
ØØ26  ØØØØ                                       ;
ØØ27  ØØØØ                 XSTART =$BA           ;(186) X1
ØØ28  ØØØØ                 XBEGIN =$BE           ;SAVE FOR XSTART
ØØ29  ØØØØ                 XFINI  =$BC          ;(188) X2
ØØ3Ø  ØØØØ                 YSTART =$BB           ;(187) Y1
ØØ31  ØØØØ                 YBEGIN =$BF           ;SAVE FOR YSTART
ØØ32  ØØØØ                 YFINI  =$BD          ;(189) Y2
ØØ33  ØØØØ                 REM    =$CØ        ;REMAINDER
ØØ34  ØØØØ                 GRAD   =$C1         ;GRADIENT
ØØ35  ØØØØ                 MODULO =$C2           ;MODULUS
ØØ36  ØØØØ                                       ;
ØØ37  ØØØØ                                       ;
ØØ38  ØØØØ                         * = $1CE8     ;START ADDRESS 74ØØ
ØØ39  1CE8                                       ;
ØØ4Ø  1CE8                                       ;
ØØ41  1CE8   A9 ØØ         LINPLT LDA #Ø
ØØ42  1CEA   85 CØ                STA REM        ;SET REMAINDER TO Ø
ØØ43  1CEC   A5 BA                LDA XSTART     ;SAVE X START COORD
ØØ44  1CEE   85 BE                STA XBEGIN
ØØ45  1CFØ   A5 BB                LDA YSTART     ;SAVE Y START COORD
ØØ46  1CF2   85 BF                STA YBEGIN
ØØ47  1CF4   4C ØØ 1D             JMP OUTPUT     ;DRAW A LINE
ØØ48  1CF7   A5 BE         ANOTHE LDA XBEGIN      ;RESTORE X COORD
ØØ49  1CF9   85 BA                STA XSTART
ØØ5Ø  1CFB   A5 BF                LDA YBEGIN     ;RESTORE Y COORD
ØØ51  1CFD   85 BB                STA YSTART
ØØ52  1CFF   6Ø                   RTS            ;BACK TO BASIC.
ØØ53  1DØØ                 ;
ØØ54  1DØØ                 ;LINE PLOT OUTPUT TESTS
ØØ55  1DØØ
ØØ56  1DØØ   A5 BA         OUTPUT LDA XSTART      ;ACC = X1
ØØ57  1DØ2   85 ØØ                STA XCOORD
ØØ58  1DØ4   A5 BB                LDA YSTART     ;ACC = Y1
ØØ59  1DØ6   85 Ø1                STA YCOORD
ØØ6Ø  1DØ8   2Ø 83 1D             JSR DDP        ; PLOT POINT ON SCREEN
ØØ61  1DØB   A5 C1                LDA GRAD
ØØ62  1DØD   FØ Ø6                BEQ TESTIT     ;GRAD Ø THUS TEST
ØØ63  1DØF   2Ø 6F 1D      VERTPT JSR VERT
ØØ64  1D12   4C ØØ 1D             JMP OUTPUT
ØØ65  1D15   A5 BA         TESTIT LDA XSTART      ;MOVE X1 INTO ACC
ØØ66  1D17   C5 BC                CMP XFINI      ;COMPARE X1 WITH X2
ØØ67  1D19   DØ ØC                BNE NEXTPT     ;BRANCH IF X1<>X2
ØØ68  1D1B   A5 BB                LDA YSTART     ;MOVE Y1 TO ACC
ØØ69  1D1D   C5 BD                CMP YFINI      ;COMPARE Y1 WITH Y2
ØØ7Ø  1D1F   FØ D6                BEQ ANOTHE     ;RETURN TO MAIN!
ØØ71  1D21   2Ø 6F 1D             JSR VERT       ;HAS TO BE VERTICAL!
ØØ72  1D24   4C ØØ 1D             JMP OUTPUT     ;BACK TO BEGINNING
ØØ73  1D27                                       ;
ØØ74  1D27                                       ;FIND GRADIENT OF LINE
ØØ75  1D27                                       ;
ØØ76  1D27   38            NEXTPT SEC             ;PREP FOR SUBTRACT
ØØ77  1D28   A5 BD                LDA YFINI
ØØ78  1D2A   E5 BB                SBC YSTART     ;ACC = Y2-Y1
ØØ79  1D2C   1Ø Ø5                BPL OVER1      ;IS Y2=>Y1?
ØØ8Ø  1D2E   38                   SEC            ;NO IT IS NOT
ØØ81  1D2F   A5 BB                LDA YSTART     ;SO DO IT OTHER WAY
ØØ82  1D31   E5 BD                SBC YFINI
ØØ83  1D33   18            OVER1  CLC             ;PREP FOR ADD
ØØ84  1D34   65 CØ                ADC REM
ØØ85  1D36   85 CØ                STA REM        ;REM = R + MOD(Y2-Y1)
ØØ86  1D38   38                   SEC            ;PREP FOR SUB
ØØ87  1D39   A5 BC                LDA XFINI
ØØ88  1D3B   E5 BA                SBC XSTART     ;ACC = X2-X1
ØØ89  1D3D   1Ø Ø5                BPL OVER2      ;IS X2=>X1?
ØØ9Ø  1D3F   38                   SEC            ;NO IT IS NOT
ØØ91  1D4Ø   A5 BA                LDA XSTART     ;DO IT THE OTHER WAY
ØØ92  1D42   E5 BC                SBC XFINI      ;ACC = X1-X2
ØØ93  1D44   85 C2         OVER2  STA MODULO      ;MODULO = MOD(X2-X1)
ØØ94  1D46   A9 ØØ                LDA #$ØØ
ØØ95  1D48   85 C1         DIVIDE STA GRAD        ;SAVE IN GRADIENT
```

```
1D4A  38                SEC              ;SET FOR SUBTRACT
1D4B  A5 CØ             LDA REM          ;ACC = MOD(Y2-Y1) + R
1D4D  E5 C2             SBC MODULO       ;ACC = MOD(Y2-Y1)+R-MOD(X2-X1)
1D4F  9Ø Ø8             BCC OUT          ;BRANCH IF Y+R<X
1D51  85 CØ             STA REM          ;Y+R = Y+R-X
1D53  A9 ØØ             LDA #$ØØ
1D55  65 C1             ADC GRAD         ;ACC = G+1
1D57  DØ EF             BNE DIVIDE       ;ALWAYS BRANCH
1D59           ;
1D59           ;CHANGE XSTART AND ALLOW IT TO BE
1D59           ;ONLY IF THE GRADIENT IS ZERO
1D59           ;
1D59  A5 BA    OUT      LDA XSTART       ;ACC = X1
1D5B  C5 BC             CMP XFINI        ;CMP X1 AND X2
1D5D  FØ A1             BEQ OUTPUT
1D5F  9Ø Ø5             BCC LDW          ;BRANCH IF X1<X2
1D61  C6 BA    LUP      DEC XSTART       ;X1 = X1-1
1D63  4C 68 1D          JMP OUTTST       ;ALWAYS BRANCH
1D66  E6 BA    LDW      INC XSTART       ;X1=X1+1
1D68  A5 C1    OUTTST   LDA GRAD
1D6A  DØ A3             BNE VERTPT
1D6C  4C ØØ 1D          JMP OUTPUT       ;ALWAYS BRANCH
1D6F           ;
1D6F           ;CHANGE YSTART USE THE GRADIENT
1D6F           ;TO DETERMIN THE NUMBER OF
1D6F           ;POINTS TO BE PLOTTED VERTICALLY
1D6F           ;
1D6F  A5 C1    VERT     LDA GRAD
1D71  FØ Ø2             BEQ NOGRAD
1D73  C6 C1             DEC GRAD         ;GRAD = GRAD -1
1D75  A5 BB    NOGRAD   LDA YSTART       ;ACC = Y1
1D77  C5 BD             CMP YFINI        ; CMP Y1 AND Y2
1D79  FØ Ø7             BEQ NOPLOT
1D7B  BØ Ø3             BCS VDOWN        ; BRANCH IF Y1>Y2
1D7D  E6 BB    VUP      INC YSTART       ;Y1=Y1+1
1D7F  6Ø                RTS              ;RETURN
1D8Ø  C6 BB    VDOWN    DEC YSTART       ;Y1=Y1-1
1D82  6Ø       NOPLOT   RTS              ;RETURN
1D83           ;
1D83           ;START OF DOUBLE DENSITY PLOTTING
1D83           ;SUBROUTINE AS GIVEN IN THE PET REVEALED
1D83           ;
1D83  A9 ØØ    DDP      LDA #$Ø          ;START OF DOUBLE DENSITY
1D85  85 B9             STA ERROR        ;PLOT ROUTINE.
1D87  85 B8             STA BINOFF
1D89  85 Ø2             STA YCOORD+1
1D8B                                     ;TEST IF YCOORD > 49
1D8B  A5 Ø1             LDA YCOORD
1D8D  C9 32             CMP #5Ø
1D8F  9Ø Ø2             BCC YOK
1D91  E6 B9             INC ERROR
1D93                                     ;TEST IF X > 79
1D93  A5 ØØ    YOK      LDA XCOORD
1D95  C9 5Ø             CMP #8Ø
1D97  9Ø Ø2             BCC XOK
1D99  E6 B9             INC ERROR
1D9B                                     ;TEST FOR OUT OF RANGE
1D9B  A5 B9    XOK      LDA ERROR
1D9D  FØ Ø1             BEQ INVERT       ;ERROR IS ZERO :CONT
1D9F  6Ø                RTS
1DAØ                                     ;INVERT THE SCREEN Y COORD
1DAØ                                     ;SAVE BOTTOM BIT OF X IN BINOFF
1DAØ  46 ØØ    INVERT   LSR XCOORD       ;DIVIDE BY 2
1DA2  26 B8             ROL BINOFF       ;SAVE CARRY IN BITØ
1DA4                                     ;SAVE BOTTOM BIT OF Y IN BINOFF
1DA4  46 Ø1             LSR YCOORD       ;DIVIDE BY 2
1DA6  26 B8             ROL BINOFF       ;SET BIT1 IN BINOFF
1DA8                                     ;MULTIPLY YCOORD BY 4Ø AND ADD
1DA8                                     ;SCREEN BASE ADDRESS
1DA8  Ø6 Ø1             ASL YCOORD       ;*2=2
1DAA  Ø6 Ø1             ASL YCOORD       ;*2=4
1DAC  Ø6 Ø1             ASL YCOORD       ;*2=8
1DAE  A5 Ø1             LDA YCOORD       ;MOVE TO ACC.
1DBØ  Ø6 Ø1             ASL YCOORD       ;*2=16
1DB2  26 Ø2             ROL YCOORD+1     ;BUMP HI IF CARRY
1DB4  Ø6 Ø1             ASL YCOORD       ;*2=32
1DB6  26 Ø2             ROL YCOORD+1
```

```
0173  1DB8  18                  CLC
0174  1DB9  65 01               ADC YCOORD       ;ADD 32+8 = 40
0175  1DBB  85 01               STA YCOORD
0176  1DBD  A5 02               LDA YCOORD+1
0177  1DBF  69 80               ADC #$80         ;ADD HIGH BYTE SCREEN ADDRESS
0178  1DC1  85 02               STA YCOORD+1
0179  1DC3                                       ;
0180  1DC3                                       ;EXPAND BINOFF
0181  1DC3                                       ;
0182  1DC3  A6 B8               LDX BINOFF
0183  1DC5  A9 01               LDA #$1          ;SET BINOFF TO 1
0184  1DC7  85 B8               STA BINOFF
0185  1DC9  E0 00       EXP     CPX #$0           ;IS IT THE END
0186  1DCB  F0 05               BEQ ENDEXP       ;YES - THEN EXIT
0187  1DCD  06 B8               ASL BINOFF       ;MULTIPLY BY 2
0188  1DCF  CA                  DEX
0189  1DD0  90 F7               BCC EXP          ;ALWAYS BRANCH BACK
0190  1DD2                                       ;
0191  1DD2                                       ;LOAD CHARACTERER FROM SCREEN
0192  1DD2                                       ;
0193  1DD2  A4 00       ENDEXP  LDY XCOORD
0194  1DD4  B1 01               LDA (YCOORD),Y
0195  1DD6                                       ;
0196  1DD6                                       ;CHECK TO SEE IF DDP ALREADY
0197  1DD6                                       ;
0198  1DD6  A2 0F               LDX #$0F         ;START AT END OF TABLE
0199  1DD8  DD FC 1D    MOREC   CMP TABLE,X       ;DO THEY MATCH?
0200  1DDB  F0 03               BEQ FOUND        ;YES - THEN EXIT
0201  1DDD  CA                  DEX              ;NO - CHECK NEXT ENTRY
0202  1DDE  D0 F8               BNE MOREC        ;JUMP BACK IF >0
0203  1DE0  A5 B7       FOUND   LDA AORD          ;ADD OR DELETE?
0204  1DE2  D0 06               BNE ERASPT       ;ERASE THE POINT
0205  1DE4                                       ;
0206  1DE4                                       ;ADD POINT TO SCREEN
0207  1DE4                                       ;
0208  1DE4  8A          ADDPT   TXA
0209  1DE5  05 B8               ORA BINOFF       ;ADD POINT TO CURRENT
0210  1DE7  AA                  TAX              ;MOVE NEW VAL TO X REG.
0211  1DE8  10 0A               BPL OUTDDP       ;NOW PRINT IT
0212  1DEA                                       ;
0213  1DEA                                       ;ERASE POINT FROM SCREEN
0214  1DEA                                       ;
0215  1DEA  A5 B8       ERASPT  LDA BINOFF
0216  1DEC  49 FF               EOR #$FF         ;PERFORM EXCLUSIVE OR
0217  1DEE  85 B8               STA BINOFF       ;TO ERASE POINT
0218  1DF0  8A                  TXA
0219  1DF1  25 B8               AND BINOFF       ;ADD TO CURRENT VALUE
0220  1DF3  AA                  TAX              ;MOVE TO X REG FOR LOOKUP
0221  1DF4  BD FC 1D    OUTDDP  LDA TABLE,X
0222  1DF7  A4 00               LDY XCOORD
0223  1DF9  91 01               STA (YCOORD),Y
0224  1DFB  60                  RTS               ;END OF SUBROUTINE
0225  1DFC  20          TABLE   .BYTE $20,$7E,$7B,$61
0225  1DFD  7E
0225  1DFE  7B
0225  1DFF  61
0226  1E00  7C                  .BYTE $7C,$E2,$FF,$EC
0226  1E01  E2
0226  1E02  FF
0226  1E03  EC
0227  1E04  6C                  .BYTE $6C,$7F,$62,$FC
0227  1E05  7F
0227  1E06  62
0227  1E07  FC
0228  1E08  E1                  .BYTE $E1,$FB,$FE,$A0
0228  1E09  FB
0228  1E0A  FE
0228  1E0B  A0
0229  1E0C                      .END

ERRORS = 0000
```